View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Universidade de Lisboa: Repositório.UL

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E ENERGIA

Photovoltaic Forecasting with Artificial Neural Networks

André Gabriel Casaca de Rocha Vaz

Dissertação

Mestrado Integrado em Engenharia da Energia e do Ambiente

2014

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E ENERGIA

Photovoltaic Forecasting with Artificial Neural Networks

André Gabriel Casaca de Rocha Vaz

Dissertação de Mestrado em Engenharia da Energia e do Ambiente

Trabalho realizado sob a supervisão de

Prof. Doutor Wilfried van Sark (Utrecht University)

Prof. Doutor Miguel Centeno Brito (FCUL)

2014

ACKNOWLEDGMENTS

I would like to show my greatest appreciation to Dr. Miguel Centeno Brito for the extraordinarily good advices, constant support and availability, and constructive comments. Without his guidance and encouragement, this dissertation would not have materialized. His insights and flexibility for open sharing of ideas were invaluable to me.

I would like to express my gratitude to Dr. Wilfried van Sark for encouraging, helping and guiding me in exploring the subject of this dissertation. In addition, a special thanks to MSc Boudewijn Elsinga for providing me valuable datasets and for being always so flexible and available whenever more information was necessary. Furthermore, the many meetings and discussions were very helpful and I would like to thank them for all the comments and suggestions.

I would like to thank my girlfriend Mariana and my parents for their support, encouragement and patience. Moreover, special thanks to my two elder brothers not only for the personal support but also for their help and advices with scientific aspects.

Last but not least I am thankful for the support that my closest friends have shown during the progress of this dissertation.

RESUMO

São necessários esforços adicionais para promover a utilização de sistemas de produção de energia fotovoltaica conectados à rede como uma fonte fundamental de sistemas de energia elétrica, em níveis de penetrações mais elevados. Nesta tese é abordada a variabilidade da geração elétrica por sistemas fotovoltaicos e é desenvolvida com base na premissa de que o desempenho e a gestão de pequenas redes elétricas podem ser melhorados quando são utilizadas as informações de previsão de energia solar. É implementado um sistema de arquitetura de rede neuronal para o modelo auto-regressivo não-linear com variáveis eXógenas (NARX) utilizando, não só, dados meteorológicos locais, mas também medições de sistemas fotovoltaicos circunjacentes. Diferentes configurações de entrada são otimizadas e comparadas para avaliar os efeitos no desempenho do modelo para previsão. A precisão das previsões revelou melhoria quando lhe são adicionadas informações de sistemas fotovoltaicos circunjacentes. Após ser selecionada a configuração de entrada da rede com o melhor desempenho, são testadas previsões com várias horas de antecedência e comparadas com o modelo da persistência, para verificar a precisão do modelo na previsão de diferentes horizontes temporais de curto prazo. O modelo NARX superou, claramente, o modelo de persistência, resultando num RMSE de 3,7% e de 4,5% aquando da antecipação das previsões de 5min e 2h30min, respetivamente.

Palavras-chave: Fotovoltaico, Redes Neuronais Artificiais, Modelo NARX, Previsão de Séries Temporais

ABSTRACT

Additional efforts are required to promote the use of grid-connected photovoltaic (PV) systems as a fundamental source in electric power systems at the higher penetration levels. This thesis addresses the variability of PV electric generation and is built based on the premise that the performance and management of small electric networks can be improved when solar power forecast information is used. A neural network architecture system for the Nonlinear Autoregressive with eXogenous inputs (NARX) model is implemented using not only local meteorological data but also measurements of neighbouring PV systems. Input configurations are optimized and compared to assess the effects in the model forecasting performance. The added value of the information of the neighbouring PV systems has demonstrated to further improve the prediction accuracy. After selecting the input configuration with the best network performance, forecasts up to several hours in advance are tested to verify the model forecasting accuracy for different short-term time horizons and compared with the persistence model. The NARX model clearly outperformed the persistence model and yielded a 3.7% and a 4.5% RMSE for the anticipation of the 5min and 2h30 forecasts, respectively.

Keywords: Photovoltaic, Artificial Neural Network, NARX model, Time Series Forecast

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

AC	Alternating current
ACF	Autocorrelation function
AIC	Akaike's information criterion
ANN	Artificial neural network
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BIC	Bayes information criterion
CPV	Concentrated Photovoltaic
CSP	Concentrated Solar Power
CV	Coefficient of Variation
DC	Direct Current
EBP	Error Backpropagation
GHG	Greenhouse Gas
KNMI	Royal Netherlands Meteorological Institute
MA	Moving Average
MAE	Mean Absolute Error
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NAN	Not A Number
NARX	Nonlinear Autoregressive with eXogenous
NWP	Numerical Weather Prediction
PACF	Partial Autocorrelation Function
PV	Photovoltaic
RMSE	Root Mean Squared Error
SSE	Sum Squared Error
TDL	Tapped Delay Line
TDNN	Time Delay Neural Network

1. INTRODUCTION

1.1. Introduction

The world's electricity consumption is growing exponentially as a consequence of world population growth and increasing per capita demand. As a result, primary energy demand will increase and, without appropriate measures, an increase in energy related greenhouse gas (GHG) emissions is expected with severe effects on our climate. However, there are development paths that allow GHG concentrations to stabilize, such as the replacement of fossil fuels (whose reserves are being rapidly depleted) by renewable energy resources, for instance, solar power. Thus, not only a decrease of the GHG emissions but also a reduction of the energy dependence is possible to establish (Heimo, Sempreviva, Kuik, Gryning, 2012).

Nowadays, electrical grids are mostly centralized, transferring power between big power plants towards end users; however, decentralized production units are expected to increase significantly. Approaches to increase electricity transfers amongst grids at different levels and penetration of renewable energies may provide a more efficient grid management. The challenge for electrical grid operators is to synchronize, continuously, the demand with energy supply.

Accordingly, as global demand for renewable energy is increasing, the economic and technical issues of solar power penetrations into the power grid must be addressed. The flat-panel PV, Concentrated Solar Power (CSP), and concentrated PV (CPV) systems are considered the most liable sources of solar energy technology to compete with fossil fuel energy production in the near future. However, natural variability of the solar resource, seasonal deviations in production and the high cost of energy storage raises concerns regarding reliability and feasibility of these systems. Moreover, solar plants usually have the support of generators for periods of high variability, increasing the costs with personnel and financially (Inman, Pedro, & Coimbra, 2013).

Unlike conventional power sources, future electricity supply cannot be precisely planned beforehand. This is due to the fact that solar energy is highly dependent on weather conditions especially cloud structure and day/night cycles. Clouds can cause significant ramps in solar insolation and PV output. Therefore, integration of electricity produced by solar power systems requires accurate solar energy potential availability evaluation and several time horizons forecasts because electricity generation varies in time and, hence, energy production pattern does not always follow the load demand. To successfully integrate increased levels of solar power production while maintaining reliability is the biggest challenge for solar energy supply and makes the availability of accurate information an important necessity.

Solar forecasts on multiple time horizons play a fundamental role in storage management of PV systems, control systems in buildings, control of solar thermal power plants, as well as for the grids' regulation and power scheduling. It allows grid operators to adapt the load in order to optimize the energy transport, allocate the needed balance energy from other sources if no solar energy is available, plan maintenance activities at the production sites and take necessary measures to protect the production from extreme events.

Depending on the purpose, different sorts of information are needed, such as the long term historical data sets of the expected energy yield (in order to assess sites where solar power systems can possible be implemented), real-time data sets (supports and optimizes energy production management), forecasted site irradiances (supports regional power grids management), local solar resource characterization and reliable estimates on the availability of solar irradiance (to uphold socio-economic planning) and, finally, real-time datasets on weather conditions (supporting forecasting of electricity demand, since this is essential to determine prices and trading of electric power) (Espinar, Aznarte, Girard, Moussa & Kariniotakis, 2010).

1.2. Motivation and Goals

Accurate solar forecasting methods improve the quality of the energy delivered to the grid and reduce the additional cost associated with weather dependency. The combination of these two factors has been the main motivation behind several research activities.

Although the existent solar forecast systems remain evasive, several approaches that combine atmospheric physics, solar instrumentation, machine learning, forecasting theory and remote sensing have been developed and presented promising results in the solar energy meteorology field. A recent study regarding a power forecasting system designed to optimize the scheduling of a small energy network including PV is described in (Kudo, Takeuchi, Nozaki, End, & Sumita, 2009). Using advanced communication networks and power predictions, the electric power and heat may be controlled especially for optimizing the energy flow. Also, a different study (Rikos, Tselepis, Hoyer-Klick, & Schroedter-Homscheidt, 2008) shows the influence of weather disturbances on the stability of an island micro-grid power system. They described how the stability may be improved when information on cloud cover approaching the island is available 15 minutes in advance. They also conclude that this information allows the start-up of power backup or the disconnection of less critical loads.

Some methods behave better for shorter time horizons and others perform better for longer time horizons. Usually, the time horizon is divided in long-term (6 hours up to days ahead) and short-term (up to 6 hours ahead). The day-ahead forecasts are required by about noon for each hour of the next day, whereas, for example in California, the intra-day ahead forecasts have to be submitted 105 minutes prior to each operating hour and at the same time have to provide advisory forecasts for the 7 hours after the operating hour (Pelland, Remund, Kleissl, Oozeki, De & Brabandere, 2013). Numerical weather forecasts, times series approaches, neural networks, use of satellite and total sky images are amongst the most used methodologies.

Numerical weather forecasts (NWP) are a common strategy for long time horizons of more than 6 hours forecasts. Basically, NWP predicts the weather by using current conditions as input into mathematical models and it has been used to forecast solar irradiance for up to several days in (Hammer, Heinemann, Hoyer-Klick, Lorenz, Mayer, & Schroedter-Homscheidt, 2007), (Lorenz, Hurka, Karampela, Beyer, & Schneider, 2008) and (Remund, Perez, & Lorenz, 2008). However, this model does not have the spatial or temporal resolution for a detailed mapping of small scale features and cannot predict how a certain solar panel is affected by cloud fields.

However, there are also classic approaches, as the times series approach, that traditionally forecast solar energy based on the time series of weather conditions and solar energy. Regarding long-term forecasting, the models used in (Bie & Musikowski, 2008), (Brinkworth, 1977) and (Puri, 1978) are based on weather station data and climate time series. On the other hand, for a time scale smaller than a day, information about cloud cover is necessary. In (Bacher, Madsen, Nielsen, & Plads, 2009) and (Dazhi, Jirutitijaroen, & Walsh, 2012) Autoregressive Models (AR), Moving Averages (MA) and Autoregressive Models Moving Averages (ARMA) are used to model linear dynamics structures and forecast hourly solar irradiance times series using cloud index.

Given the limitations of the basic models previously presented, research has been done in nonlinear models that show more flexibility in capturing the data underlying characteristics (Artificial Neural Networks). In order to fit the network, training of the model is involved over the known input and output values. The authors in (Zeng & Qiao, 2011) show that an artificial neural network-based model for short-term solar power prediction outperforms the AR model.

The use of total sky and satellite images stand out as models for very short-term forecasting because incorporate information on the actual atmospherics state by applying image processing and cloud tracking techniques. In (Jayadevan, Rodriguez, Lonij, & Cronin, 2012) the authors analyzed digital images taken with a ground-based sun tracking camera and discuss statistics of ramp rates and duration of cloud induced intermittencies; Also, (Marquez & Coimbra, 2012) describes several sky image processing techniques relevant to solar forecasting, including velocity field calculations, spatial transformation of the images, and cloud classification.

Furthermore, satellite imagery is based on the premise that clouds reflect light from earth into the satellite, leading to the detection and ability to calculate the amount of light transmitted. The low spatial and temporal resolution causes satellite forecasts to be less accurate than total sky imagery. However, in the 1 to 5 hours range satellite imagery has a better forecasting accuracy. In (Hoff & Perez, 2012), the authors suggested that satellite-based irradiance has an annual error comparable to ground sensors and is suitable to provide the data required to perform high penetration PV studies.

As illustrated in Figure 1, for the very short-term time horizon, from minutes up to a few hours, time series models using on-site irradiance measurements or power data as input are adequate. Moreover, regarding Intra-hour forecasts of clouds and irradiance with a high spatial and temporal resolution total sky images are the best option. Forecasts based on cloud motion vectors from satellite images show good performance for a temporal range of 30 minutes to 6 hours. Finally, grid integration of PV power mainly requires forecasts up to 2 days ahead or even beyond and these forecasts are based on NWP models.

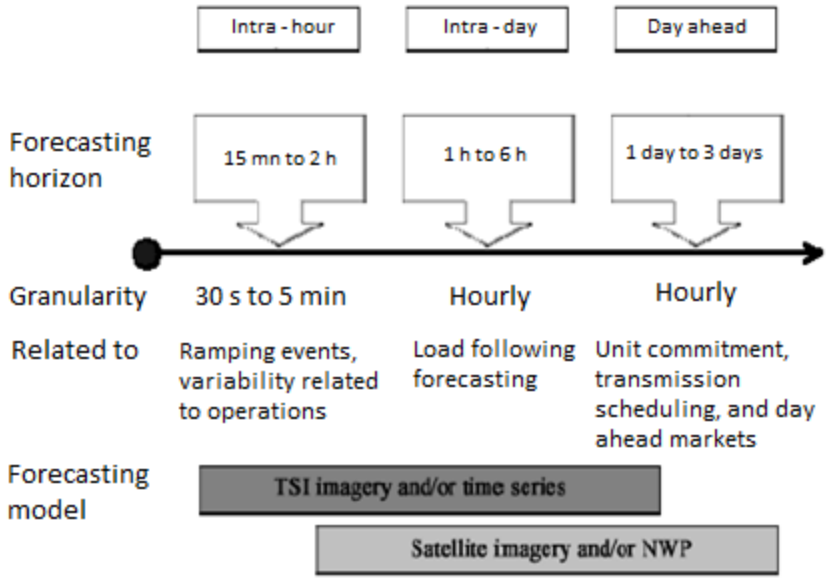

Figure 1 - Relation between forecasting horizons, forecasting models and the related activities. (Diagne, David, & Boland, 2012)

Figure 1 depicts the temporal and spatial resolution of different forecasting models. ARIMA models present significant reliability in the statistical forecasting model range. However, one verifies in Figure 2 that the persistence[1] model often achieves better accuracy than ARIMA-type models for real time forecasts. The choice of the model depends critically on the horizon of forecast. At higher frequency, short-term patterns dominate and Artificial Neural Networks demonstrated good results. (Diagne e al., 2012)

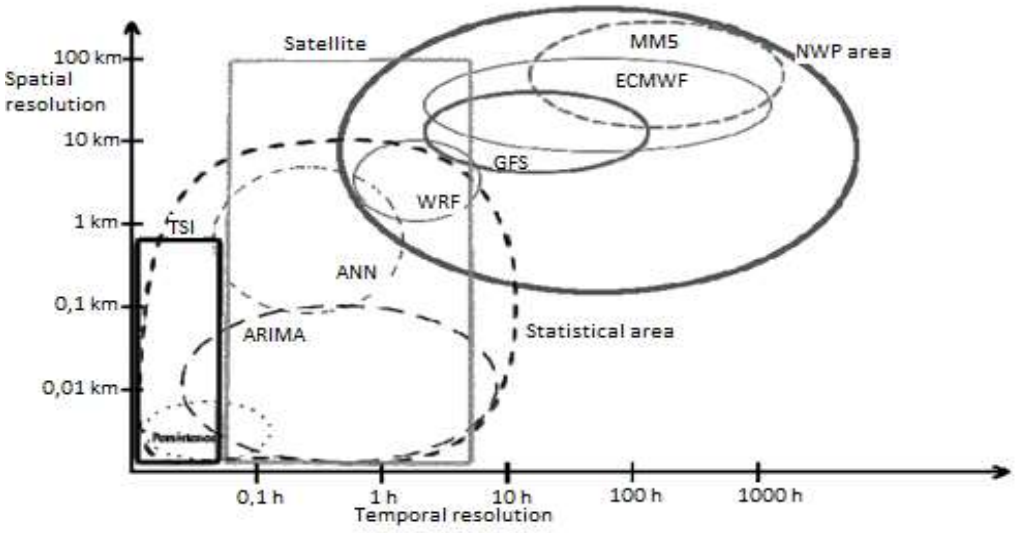

Figure 2 - Classification of the forecasting models (Temporal Resolution vs Spatial Resolution). (Diagne et al., 2012)

The present work intends to use an artificial neural network (ANN) model to capture the short-term ramping patterns caused by cloud formations and to forecast a PV system power output up to 1 day ahead. Moreover, using different input combinations, we want to assess whether or not solar power forecasts can be improved by knowing beforehand the power output of other neighbouring grid-connected PV systems and local meteorological information.

1.3. Thesis Scope

In section 2, an introduction to the architecture and relevant variations of neural networks is presented. Furthermore, typical photovoltaic systems are described and the current state of the art of solar forecasting with artificial neural networks is reviewed. Additionally, several functions that can evaluate the quality of the neural network predictions are indicated. Section 3 describes the design and implementation of the NARX model using neural networks. In section 4, the results of the experiments and tests are presented and thoroughly discussed. Finally, in section 5, the conclusion of the work and future research expectations in the solar forecasting domain are elaborated.

[1] Simple model that meets the definition: $X_{n,y} = X_{n-k,y}$ where k denotes the lag (k = 1,2,3,..,m).

2. PHOTOVOLTAIC FORECASTING

2.1. Artificial Neural Networks

A thorough understanding of the architecture of neural networks is important to avoid disappointing results and, thus, identify and establish better parameters to improve the network performance. Therefore, this section describes the fundamentals of artificial neural networks.

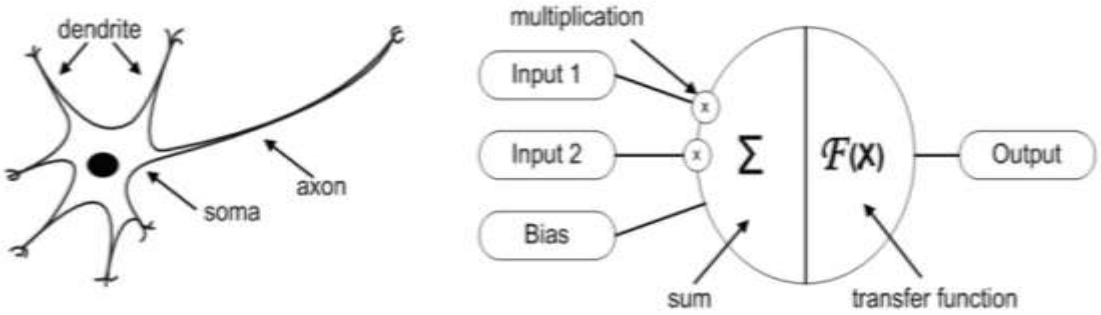

Figure 3 - Biologic and artificial neuron designs. (Krenker, Bešte, & Kos, 2011)

The design and functionalities of the artificial neuron derive from the observation of the complex biological neuron in which distributed information is processed in parallel by mutual dynamical iterations of the neuron. Accordingly, there are some similarities between the biological neural network and the artificial neural network and one can verify it in Figure 3. In the biological neuron the information comes into the neuron via dendrite, soma processes it and passes it on via axon. Similarly, in the artificial neural network the information comes from the inputs that are weighted. Consequently, in the artificial neural body the weighted inputs and bias are summed and processed with a transfer function. After being processed, the information is passed via outputs.

Different learning rules can be chosen and applied, and, consequently, the weights and bias are adjustable parameters so that the neuron input/output achieves a specific end. In any artificial neural network model, it is important to consider the structure of the nodes, topology of the network and the learning algorithm. Therefore a broader view of the mathematical and fundamentals and algorithms will be presented.

2.1.1. Artificial Neural Networks: Definitions and Properties

2.1.1.1. Single Input-Neuron

A neural network consists of simple processing units, the neuron, and directed, weighted connections between those neurons (Figure 4). The inputs channels have an associated weight, which means that the incoming information x_i is multiplied by the corresponding weight w_i. The network input is the result of the latter process, so-called propagation function. Here, the strength of a connection between two neurons i *and* j is a connecting weight and illustrated by w_{ij}. (Kriesel, 2005)

These connecting weights can be inhibitory or excitatory and by being connected with the neurons, data are transferred. Figure 4 illustrates the single-input neuron. wp is formed after the scalar input p is multiplied by the scalar weight w. Consequently, n, often referred to as the net input is processed into a transfer or activation function f. The latter process gives the scalar neuron output a. Thus, the output is a function of the particular activation function chosen and the bias. The latter is similar to a weight, albeit it has a constant input of 1. This bias term is used by the neuron to generate an output signal in the absence of input signals.

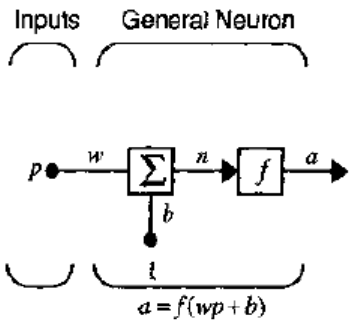

Figure 4 - Single-Input Neuron. (Beale, Hagan, & Demuth, 2013)

2.1.1.2. Neuron with vector input

The simple neuron previously shown can be extended to handle inputs that are vectors. The concept is the same as before: the individual elements in a neuron with a single R-element input, vector $p_1, p_2, \dots, p_R$ are multiplied by weights $w_{1,1}, w_{1,2}, \dots w_{1,R}$ and then fed to the summing junction. The sum of the weighted values is Wp, the product of the matrix W and the vector p. In order to form the network input n, there is a bias b in the neuron which is summed with the weighted inputs. Consequently, the network input n is the argument of the activation function f,

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b \quad (1)$$

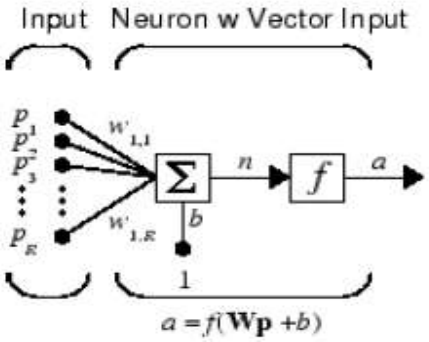

Figure 5 - Neuron with vector input. (Beale et al., 2013)

2.1.1.3. Transfer function

Transfer function or activation function controls the amplitude of the output of the neuron and is based on the neurons reactions to the input values and depends on the level of activity of the neurons (activation state). This premise is founded on the biological model, where every neuron is, at all times, somewhat active. Essentially, neurons are activated when the network input exceeds the uniquely maximum gradient assigned value of the activation function, known as threshold. Accordingly, near the threshold value the activation function has a rather sensitive reaction. The activation function is dependent of the previous activation state of the neuron and the external input and is defined as

$$a_j(t) = f_{act}\left(net_j(t), a_j(t-1), \Theta_j\right) \tag{2}$$

This equation demonstrates how the network input net, previous activation state $a_j(t-1)$ and the influence of the threshold Θ_j, is transformed into a new activation state $a_j(t)$. It must be emphasized that though the threshold values are different for each neuron, the activation function embraces all neurons.

Two of the most commonly used activation functions in neural networks are the logistic and hyperbolic tangent function. Both functions are used because of the simplicity in finding its derivatives. Usually, these functions are applied in the hidden layer of the network.

The logistic function, $sigmoid\ (x) = \frac{1}{(1+e^{-x})}$ takes the input with any value between plus and minus infinity and maps the output to the range values (0, 1). The hyperbolic tangent: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ also takes the input with any value between plus and minus infinity and squashes the output into the range -1 to 1. The selection of the activation function provides nonlinear limits to the hidden neurons and influences the performance of the networks. To avoid bad performances, one usually preprocesses the input data, for example, by normalizing the data.

Another relevant function is the linear function $f(x) = x$, where the inputs and outputs range from minus infinity to plus infinity, which it is generally used in the output layer of the network.

2.2. Neural Networks Architecture

The neuron is a nonlinear, parameterized function of its input. The configuration of the nonlinear functions of two or more neurons is a neural network. The next sections introduce the different neural networks classes: feedfoward networks and recurrent (feedback) networks.

2.2.1. Feedforward Neural Networks

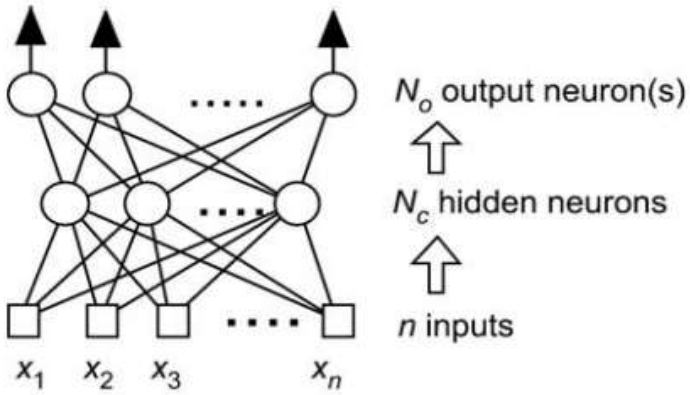

Figure 6 - Feedfoward neural network with n inputs, a layer of N_c hidden neurons, and N_0 output neurons. (Krenker et al., 2011)

A Feedforward neural network is a nonlinear function of its inputs, which is the composition of the functions of its neurons. As Figure 6 illustrates, the information runs through the connected neurons only in the forward direction, from inputs to outputs. Graphically, the vertices are the neurons and the edges are the connections; these types of networks do not have back-loops. Obviously, the term connection is taken metaphorically because the computations by each neuron are implemented as software programs.

2.2.2. Multilayer Networks

Most neural networks applications require the use of multilayer networks with a similar topology as the one in Figure 6, which illustrates how the network computes N_0 functions of the input variables of the network; each output is a nonlinear function of the nonlinear functions computed by the hidden neurons. In other words, in the Feedforward neural network the N_0 nonlinear functions are computed based on the previous computation of the N_c functions computed by the hidden neurons.

Feedforward neural networks are considered static neural networks models, that is, models applicable to processes where the setting for each piece are determined up front, and are not altered for that piece using feedback during the process. (Coit, Jackson, & Smith, 1997)

Furthermore, Feedforward multilayer networks that use sigmoid nonlinearities are also designated as Multilayer Perceptron (MLP) networks. The following equations and Figure 7 present the structure and calculations required to generate outputs of single multilayers Feedforward artificial neural networks.

$$n_1 = F_1(w_1 x_1 + b_1) \tag{3}$$

$$n_2 = F_2(w_2 x_2 + b_2) \tag{4}$$

$$n_3 = F_2(w_2 x_2 + b_2) \tag{5}$$

$$n_4 = F_3(w_3 x_3 + b_3) \tag{6}$$

$$m_1 = F_4(q_1 n_1 + q_2 n_2 + b_4) \tag{7}$$

$$m_2 = F_5(q_3 n_3 + q_4 n_4 + b_5) \tag{8}$$

$$y = F_6(r_1 m_1 + r_2 m_2 + b_6) \tag{9}$$

$$y = F_6\begin{bmatrix} r_1(F_4[q_1 F_1[w_1 x_1 + b_1] + q_2 F_2[w_2 x_2 + b_2]] + b_4 + \cdots \\ \ldots + r_2(F_5[q_3 F_2[w_2 x_2 + b_2] + q_4 F_3[w_3 x_3 + b_3] + b_3]) + b_6 \end{bmatrix} \tag{10}$$

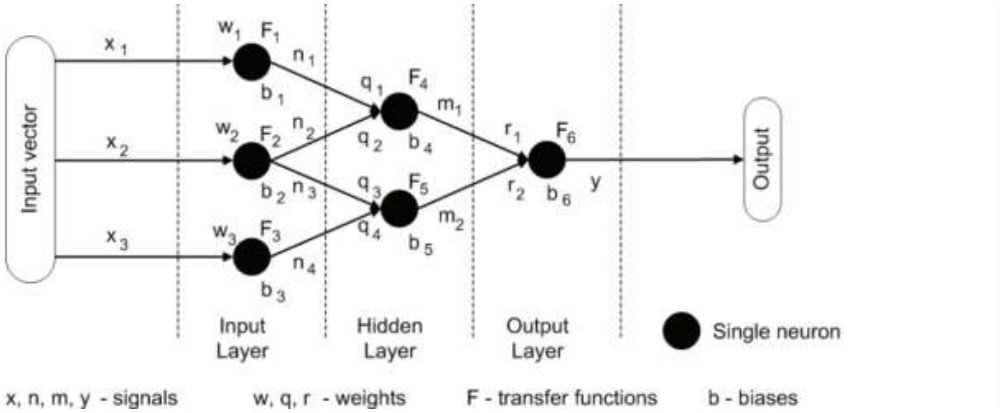

Figure 7 - Multilayer artificial neural network. (Krenker et al., 2011)

2.2.3. Multilayer Perceptron and the hidden nodes

The Multilayer Perceptron is one of the most important models in the artificial neural networks domain. For prediction purposes, data is presented to the MLP as a sliding window over the time series observations. The task of the MLP is to model the underlying generator of the data during training, so that a valid forecast is made when the trained neural network is subsequently presented with a new input vector value. (Bramer, 2006)

The inherent capability of the three-layer network structure to carry out any arbitrary input-output mapping highly qualifies the MLP networks for efficient time series forecasting. When examples of the observation data are trained, the networks can learn the characteristic features "hidden" in the examples of the collected data and even generalize the knowledge learnt. (Palit & Popovic, 2005)

The hidden layer nodes are fundamental, albeit there is a large controversy regarding the number of nodes and hidden layers that are necessary to guarantee a good network performance. Due to the fact that no theoretical answer exists, heuristics processes are applied and have been generating some rules of thumb depending on the task. Usually, one hidden layer is enough to characterize the task because several hidden layers may generate unwanted complexity to the problem. (Coit et al.,1997)

In general, one should select enough hidden neurons to generate a solution to a task. However, if the group of patterns of input available is not enough, it is not recommended to have an amount of nodes that generates an estimation of the weights that is not trustworthy.

2.2.4. Recurrent Neural Networks

Recurrent Neural Networks are similar to Feedfoward neural networks but with no limitations regarding back-loops, that is, the network exhibits cycles (Figure 8). Therefore, information may be transmitted both forward and backwards. Consequently, an internal state of the network is created displaying a dynamic temporal behaviour. (Krenker et al., 2011)

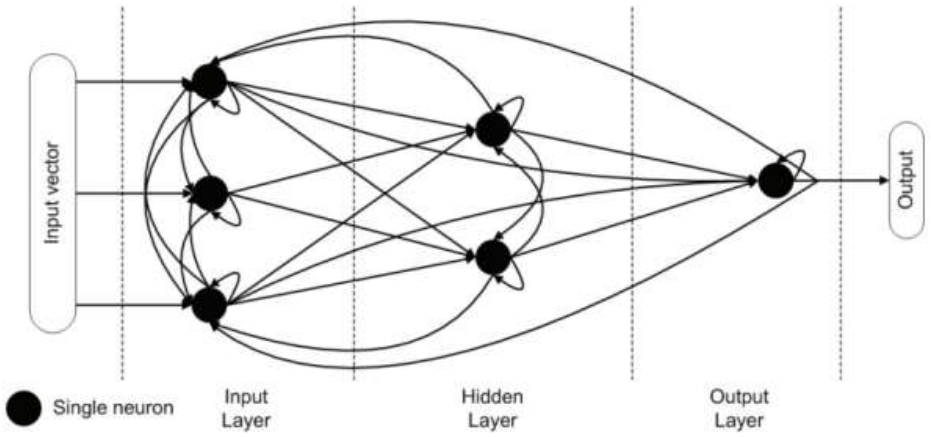

Figure 8 - Fully recurrent artificial neural network. (Krenker et al., 2011)

Given the fact that the output of a neuron cannot be a function of itself but can be a function of past values, these architectures require *time* to be explicitly taken into consideration. The ordinary framework applied to recurrent networks is the discrete-time system, which is described mathematically by recurrent equations.

These equations are discrete-time equivalents of continuous-time differential equations. Therefore, besides being assigned a parameter as in Feedforward neural networks, a delay is assigned to each connection of a recurrent neural network (this delay can be made equal to zero). Each delay is a numeric value multiple of an elementary time that is considered as a time unit.

Essentially, a discrete-time recurrent neural network follows a set of non-linear discrete-time recurrent equations, not only through the neurons functions configuration but also through the time delays associated to its connections.

2.3. Dynamic Driven Recurrent Networks

Most dynamical systems involve an autonomous part and a part governed by external force that usually is difficult to identify or noisy. Forecasting deals with dynamic models whose inputs and outputs are related through differential equations, or, for discrete-time systems, by recurrent equations. Recurrent networks with global feedback will be discussed, which is relevant for the scope of this thesis. For a thoroughly understanding of recurrent networks with local feedback, (Haylin, 1999) is suggested.

Considering the typical design of the multilayer networks previously shown, applying the global feedback can take a variety of arrangements. Global feedback can either be in a form of output neuron to the input layer or from the hidden neuron to the input layer. Other architectural layouts for recurrent networks exist, for instance, for multilayer networks with more than one hidden layer; however, those are not relevant for the current work and will not be discussed in detail.

Pertinent to this work is the discussion of recurrent networks used as input-output mapping networks. Basically, in this situation, an external input is applied and the recurrent network has a temporary response. Consequently, the recurrent network is considered as dynamically driven recurrent network. This characteristic enables recurrent networks to acquire state representations, which are fundamental for applications such as nonlinear predictions and modelling. In section 2.7, the recent use of neural networks for forecasting purposes is thoroughly discussed.

2.3.1. Input-Output Recurrent Model

The input-Output recurrent model, with a design that follows the typical multilayer perceptron, is illustrated in Figure 9. One can notice that the model has a single input that is applied to a tapped-delay-line (TDL) memory of q elements. A delay line tap extracts a signal output from somewhere within the delay line and usually sums with other taps to form an output signal. Moreover, via another TDL memory with q units, the single output is also fed back to the input. Thus, the contents from both TDL memories are fed to the input layer of the multilayer perceptron.

In Figure 9, $u(n)$ denotes the present value of the model input and $y(n+1)$ corresponds to the value of the model output. Accordingly, one may understand that the output is one time unit ahead of the input. Hence, the present and past values of the input, which are exogenous inputs generated from outside the network, and delayed values of the output, on which the model output is regressed, are the data window of the signal vector applied to the input layer.

This recurrent network described above and shown in Figure 9 is also referred as nonlinear autoregressive with exogenous inputs (NARX) model (Haylin, 1999).

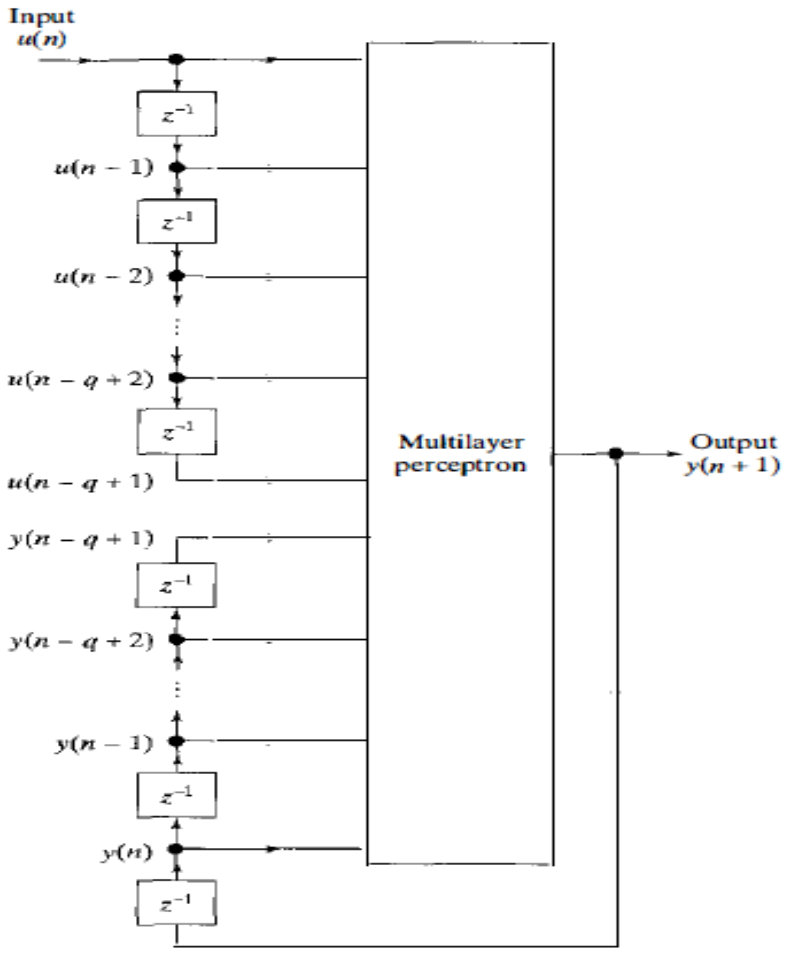

Figure 9 - Nonlinear autoregressive with exogenous inputs (NARX) model. (Haylin, 1999)

$$y(n+1) = F(y(n), \dots, y(n-q+1), u(n), \dots, u(n-q+1)) \quad (11)$$

Equation 11 demonstrates the dynamic behaviour of the NARX model, where F is a nonlinear function of its arguments. The two delay line memories in the model are generally different, albeit they can have the same q size.

2.4. Training a Neural Network

A key aspect in the implementation of artificial neural networks is the training. This process must be well designed so that the network successfully learns a task. However, one should understand that a precise definition of training is difficult to achieve because there is no direct approach on how to do this (Jain & Mao, 1996). This learning process consists in the adjustment of the weights under some learning rules. Essentially, the free parameters from a network are adapted, through a stimulation process. When a group of patterns is presented, the network typically learns

the connection weights and the performance is improved by iteratively updating the weights. The network learns to recognize the pattern inherent to the training signals.

Though the learning process poses some issues, the ability to automatically learn from examples and learn underlying rules, such as the input-output relationships, makes the neural networks more attractive than traditional systems. Theoretically, the network must approach the global minimum of the objective function, that is, the error function will steadily decrease until the minimum error has been reached. If this is achieved, and no further decrease of the error function is necessary, the training process must be stopped. In practice, the network training can require several training trials with various initial weight values in order to find this global minimum. After each training run, an evaluation and comparison between the training results and the results achieved in the previous run allow us to select the best run.

The design of the training or learning process has to consider the model of the environment in which a neural network works. Thus, one has to distinguish which information is available to the network. Moreover, it is essential to understand how the network weights are updated, i.e. the learning rules that the updating process must follow.

There is not a unique algorithm for the design of neural networks and the learning process of the neural networks can either be classified as supervised or unsupervised training. Essentially, these classifications differ in the existence or not of an external agent (supervisor) that controls the learning process in the network. Other classification criteria reside in defining if the network learns through its normal functioning (online) or if the learning assumes the unplugging of the network (offline). For an online training the weights vary dynamically when new information is shown to the system. Inversely, the networks that use offline learning have their connection weights remain fixed after the training stage.

In supervised training, for every input pattern an output is provided to the network and the external agent controls the answer that the network must generate based on a determined answer, that is, the supervisor compares the output of the network with the expected results and determines the amount of modification that must be applied in the weight. Accordingly, weights are determined so that the result is as close as possible to the known correct answers, i.e. the objective is to find the minimum value of the difference between the answer of the network and the correct answer.

Differently, the unsupervised training organizes patterns into categories from the underlying structure in the data or correlation between patterns in the data. With this method, the neural network is capable of self-organize because there is no information received from the environment indicating the correctness of a generated output. Basically, there is no correct answer required. The interpretation of the output of unsupervised networks depends on the structure of the network and the learning algorithm used. Sometimes the output represents the degree of similarity between the signal introduced in the network and the displayed information until then. Under certain circumstances, grouping of information (clustering) is established, where each category is set based on the correlation between the presented information.

Theoretically, there are some fundamental issues associated with learning from samples that must be considered, such as, the capacity, sample and computational complexity. The capacity refers to the functions and boundaries a network can form, that is, the quantity of patterns that may be stored. Assessing the complexity of the sample is highly important, as it determines the necessary patterns that need to be train in order to achieve a valid generalization. Finally, the computer complexity refers to the time that a chosen algorithm requires to reach an estimate solution from the trained patterns.

The experiment design for network training involves concerns regarding the network initialization training, selection of the appropriate training algorithm, formulation of training stopping criteria, etc.

2.4.1. Avoiding Overfitting

One must find the information that allows us to confirm that the maximum generalization has been reached. Figure 10 presents the case where after reaching the point of maximum generalization the network keeps learning from the training set; however, it starts to damage the related test set performance due to its overtraining. Furthermore, in Figure 10 the overfitting is caused when the validation error increases while training error decreases progressively. Reducing the number of hidden neurons is an option to avoid overfitting. (Tan, 2009)

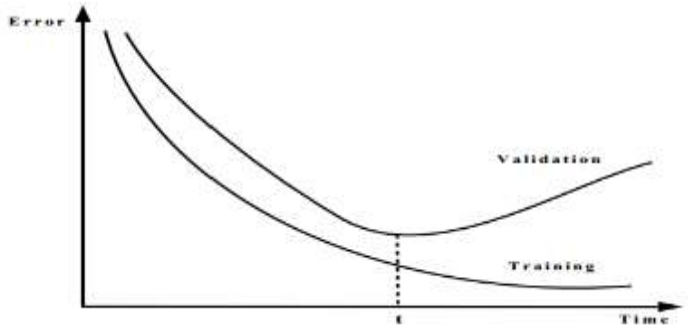

Figure 10 - Overtraining example. (Palit & Popovic, 2005)

However, in (Palit & Popovic, 2005) a better approach to solve the training termination problem based on stopping criteria was presented. They developed an automated stopping principle using a predetermined number of training steps. Ideally, the stopping strategy is the one that stops the training after the network has learnt all the problem details it has to solve. Consequently, when the training stopping achieves that stage, the network reaches the maximum *generalization***.** Thus, the minimum value has been reached and this is the point where stopping should be activated. This action is known as *early stopping*. Beyond this point, the network would be performing the so-called *network overtraining or overfitting*.

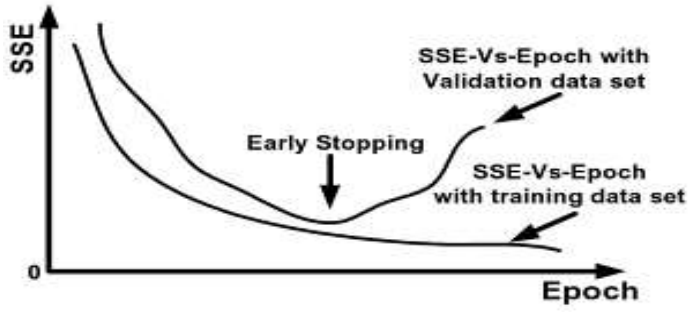

Figure 11 - Early stopping of training. (Palit & Popovic, 2005)

To prevent overfitting, in (Prechelt, 1998) the author suggested the method of early stopping with cross-validation. This method proposes the division of collected data into a training set and a test set, and for further partitioning of the training set into the estimation set and validation set. Yet, finding the exact location of the early stopping is not an easy task. Therefore, to manage the problem a stopping principle was introduced consisting in subdividing the training set into the training error E_{train} (average error per example across the training set), the test and the validation error, E_{test} and E_{val} respectively.

Both the problem of *overfitting* and the opposite problem of *underfitting* are consequences of improper training stopping. The network ability to generalize is affected and lowered by both problems and should be prevented. In the *underfitting* problem, the network trained is less complex than the task to be learnt, therefore, poorly identifies the structures within a large training data set. Inversely, when trained, a very complex network not only can extract the structures within the training set, it also extracts the embedded noise. This may pose results and predictions that are not acceptable.

The network complexity is related to the number of weights and it is determined by the prediction accuracy of the model selected. The latter depends on the number and size of weights and hidden neurons that would implement the desired prediction accuracy without performing *overfitting*. Statistically, the *underfitting* and *overfitting* are related to the statistical bias and the statistical variance they produce. The statistical bias is related to the degree of target function fitting and constrains the network complexity; however, disregards the trained network generalization. The statistical variance (deviation of network learning efficiency within the set of training data) cares about the generalization of the trained network. It is difficult to get the balance between both as the *underfitting* generates a high bias network and the *overfitting* produces a large variance.

2.4.2. Training Algorithm

One of the most significant breakthroughs for training neural networks was the development of the steepest descent algorithm, also known as error backpropagation (EBP) algorithm. For each example in the training set, the algorithm calculates the error using a predefined error function, that is, the difference between the actual and desired outputs. After that process, the error is back propagated through the hidden nodes to adjust the weights of the inputs. This procedure is completed when the network converges to a minimum error solution. Though this algorithm is widely used in neural networks, it presents some limitations, in particular slow convergence and easily traps in local minima. (Dreyfus, 2005)

When the gradient is steep, small step sizes should be taken to not rattle out of the required minima. On the other hand, for a small constant step size the training process would be very slow when the gradient is gentle. Also, the classic "error valley" can occur when the curvature of the error surface has different directions and, therefore, can result in slow convergence. However, the slow convergence of the steepest descent method can be significantly enhanced by the Gauss-Newton algorithm which is able to find adequate step sizes for each direction and can converge very fast by using second-derivatives of error function to evaluate the curvature of error surface. Yet, calculating the second-derivatives poses computational complexity.

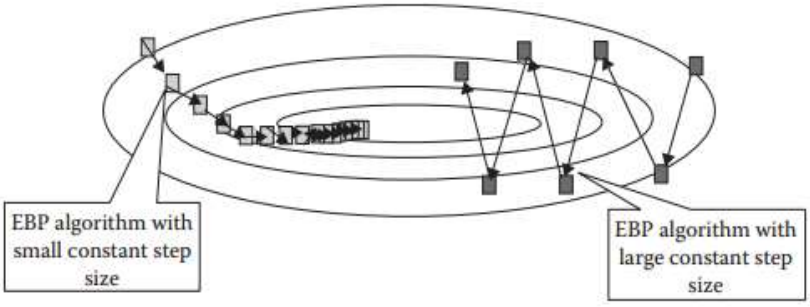

Figure 12 - Steepest descent method with different learning constants. The trajectory on the left is for small learning constant that leads to slow convergence; the trajectory on the right is for large learning constant that causes oscillation (divergence). (Yu & Wilamowski, 2010)

To overcome these problems other learning algorithms were proposed such as the Levenberg-Marquardt which is suitable for small and medium sized problems and has a fast and stable convergence when compared to other methods. It combines the steepest descent and the Gauss-Newton algorithms. It has the stability of the steepest descent method and the speed of the Gauss-Newton but it is more robust than the Gauss-Newton. The idea is to combine both training processes so that around the area with complex curvature the algorithm switches to the steepest descent algorithm, until the local curvature is adequate to complete a quadratic approximation; later, to speed up the convergence, the algorithm approximately becomes the Gauss-Newton algorithm (Yu & Wilamowski, 2010).

2.4.3. Levenberg-Marquardt Algorithm Origin

This section explains how the Levenberg-Marquardt method derived from the combination of algorithms.

2.4.3.1. Steepest Descent Algorithm

The backpropagation algorithm is used to learn the weights of a multilayer neural network and performs gradient descent to minimize the sum squared error between the network's output and a certain target value. The error is squared because its magnitude is more relevant than its sign. The total error *E* is given by the following equation

$$E(x, w) = \frac{1}{2} \sum_{p=1}^{P} e_p^2 \tag{12}$$

where P is number of training patters, x is the input vector, w the weight vector and e_p defines the training error for training pattern p. e_p is obtained by,

$$e_p = (o_i - t_i)^2 \tag{13}$$

where o_i is the network output at the i^{th} output node, t_i is the target output at the i^{th} output node. Every algorithm adjusts the weights and biases to reduce this global error.

To overcome the problem of finding global solutions to the error given the non-linearity of the error function, the algorithm is set to analyze the weight space. Therefore, it is formulated as follows:

$$w_{k+1} = w_k - \Delta w_k \tag{14}$$

where *k* is the index of iterations.

The steepest descent algorithm uses the first-order derivative of total error function to find the minima in error space. The first-order derivative of total error function E defines gradient g:

$$g = \frac{\partial E(x,w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \ \frac{\partial E}{\partial w_2} \ \cdots \ \frac{\partial E}{\partial w_n}\right]^T \tag{15}$$

Based on the definition of gradient *g*, it can be written the update rule of the steepest descent algorithm:

$$w_{k+1} = w_k - \alpha g_k \tag{16}$$

where α is the learning constant (step size).

All the elements of gradient vector would be very small with a slightly weight adjustment around the solution. Therefore, this training process is asymptotic convergence.

2.4.3.2. Newton's Method

Newton methods can be relatively slow because they explicitly use the full Hessian matrix *H*, which must be calculated and, therefore, some computational expense occurs. The Hessian matrix *H* gives the proper evaluation on the change of gradient vector with the second-order derivatives of total error function. Through several mathematical equations and using Taylor Series (Yu & Wilamowski, 2010), it can be demonstrated that:

$$-g = H\Delta w \Leftrightarrow \Delta w = -H^{-1}g \tag{17}$$

Consequently, the update rule for Newton's method is

$$w_{k+1} = w_k - H_k^{-1} g_k \tag{18}$$

Where *H* is

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_n^2} \end{bmatrix} \quad (19)$$

One is able to identify the differences between the equations of the steepest descent method and the Newton's method and notice that complementary step sizes are given by the inverted Hessian matrix.

2.4.3.3. Gauss-Newton Algorithm

Although it is rather complicated to calculate the second-order derivatives of the total error function that allow us to determine the Hessian matrix *H*, this process is essential for Newton's method because it is applied for the weight updating. To simplify the calculating process, the Jacobian matrix *J* can be introduced. The Jacobian matrix is the matrix of the first-order partial derivatives of the error function as illustrated in the following equation.

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \cdots & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,1}}{\partial w_1} & \frac{\partial e_{p,1}}{\partial w_2} & \cdots & \frac{\partial e_{p,1}}{\partial w_N} \\ \frac{\partial e_{p,2}}{\partial w_1} & \frac{\partial e_{p,2}}{\partial w_2} & \cdots & \frac{\partial e_{p,2}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{p,M}}{\partial w_1} & \frac{\partial e_{p,M}}{\partial w_2} & \cdots & \frac{\partial e_{p,M}}{\partial w_N} \end{bmatrix} \quad (20)$$

The relationship between Jacobian matrix J and the gradient vector g is shown in (Yu & Wilamowski, 2010) to be

$$g = Je \quad (21)$$

where *e* is the error vector. Moreover, it is proved in (Yu & Wilamowski, 2010) that the relationship between the Hessian matrix *H* and Jacobian Matrix *J* can be written as

$$H \approx J^T J \tag{22}$$

Consequently,

$$w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k \tag{23}$$

This equation clearly demonstrates that calculating the second-order derivatives of the total error function is not required. Thus, the Gauss-Newton algorithm has this advantage comparing to the standard Newton's method. Nonetheless, the Gauss-Newton method still presents some problems regarding the convergence for complex error space optimization just as the Newton's method. Mathematically, the $J_k^T J_k$ can pose a problem because this matrix may not be invertible.

2.4.3.4. Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm presents another approximation to the Hessian matrix in order to make sure that the matrix $J_k^T J_k$ is invertible:

$$H \approx J^T J + \mu I \tag{24}$$

where μ is the combination coefficient (always positive), and *I* is the identity matrix.

This approximation insures that the matrix H is always invertible because the elements of the main diagonal of the approximated Hessian matrix are larger than zero. Consequently, by combining equation (18) and equation (24), the update rule of Levenberg-Marquardt algorithm is

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \tag{25}$$

Hence it is demonstrated the combination between the Gauss-Newton algorithm and the steepest descent algorithm. The Levenberg-Marquardt algorithm switches between both algorithms during the training process. When μ is very small, that is, very close to zero, the Levenberg-Marquardt algorithm switches to the Gauss-Newton algorithm. On the other hand, when μ is large, the steepest descent method is used because the equation (25) approximates the equation (16). Table 1 summarizes the differences between the different training algorithms and its main features regarding speed, stability and computational complexity.

Table 1 - Characteristics of the algorithms.

Algorithms	Update Rules	Convergence	Computational Complexity
EBP	$\boldsymbol{w_{k+1} = w_k - \alpha g_k}$	Stable, slow	Gradient
Newton	$\boldsymbol{w_{k+1} = w_k - H_k^{-1} g_k}$	Unstable, fast	Gradient and Hessian
Gauss-Newton	$\boldsymbol{w_{k+1} = w_k - \left(J_k^T J_k\right)^{-1} J_k e_k}$	Unstable, fast	Jacobian
Levenberg-Marquardt	$\boldsymbol{w_{k+1} = w_k - \left(J_k^T J_k + \mu I\right)^{-1} J_k e_k}$	Stable, fast	Jacobian

One might have noticed that according to the updating rule of the Levenberg-Marquardt algorithm, if the error is decreasing, that is, the error in $k+1$ is smaller than in k, the μ coefficient can be reduced so that the influence of gradient descent part is diminished. However, if the opposite occurs, if the error increases, it is necessary to follow the gradient to look for a proper curvature for quadratic approximation and the coefficient μ is increased.

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be rather large for certain problems.

The following section introduces the photovoltaic technology and further down the use of artificial neural networks in the photovoltaic domain is discussed.

2.5. Photovoltaic Systems

The sun can be considered as the source of almost all energy on the planet, because most of the available energy is directly (sunlight) or indirectly (wind and waves) related with it. The sun's apparently ability to provide endless energy results from the process of nuclear fusion. This energy, produced in the core of the sun, is emitted as electromagnetic radiation. Though electromagnetic radiation is emitted in many useful forms, the solar cell designers are more interested in capturing the energy carried in visible light. (Stapleton & Neill, 2012)

The present section introduces typical small-scale PV systems from which information regarding the systems' energy production is collected. This section also analyses relevant factors that influence that production.

2.5.1. Photovoltaic Technology

PV cells are devices that produce electricity directly from electromagnetism radiation. These devices are made from semiconducting materials, which conduct electricity under specific conditions, so they are neither insulators nor conductors. The most common semiconductor material is silicon, which is often combined with other elements to improve its conductivity, in a process

designated as doping. Controlled quantities of specific impurity ions are added to the very pure material to produced doped semiconductors.

Impurity dopant ions of fewer valences (e.g. boron) enter the solid Si lattice and become electron acceptor sites which trap free electrons. These traps have an energy level within the band gap, but near to the valence band. The absence of the free electrons produces positively charged states called holes that move through the material as free carriers. With such electron acceptor impurity ions, the semiconductor is called p (positive) type material, having holes as majority carriers. On the other hand, atoms of great valency (e.g. phosphorus) are electron donors, producing n (negative) type material with an excess of conductions electrons as the majority carriers. (Twidell & Weir, 2006)

An electron free to move throughout the crystal is said to be in the crystal's conduction band, because free electrons are the means by which electricity flows. Both the conduction-band electrons and the holes are fundamental in the electrical behavior of PV cells. Although the generation of electrons and holes by light is the central process in the overall PV effect, it does not itself produce a current.

A PV cell contains a barrier that is set up by opposite electric charges facing one another on either side of the junction. This potential barrier selectively separates light-generated electrons and holes, sending more electrons to one side of the cell, and more holes to the other. This charge separation sets up a voltage difference between either ends of the cell, which can be used to drive an electric current in an external circuit. (Zweibel, 1982)

If we connect the n-type side to the p-type side of the cell by means of an external electric circuit, current flows through the circuit because this reduces the light induced charge imbalance in the cell. This current from the cell is inherently direct current (DC). Figure 13 illustrates the functioning of a typical PV cell.

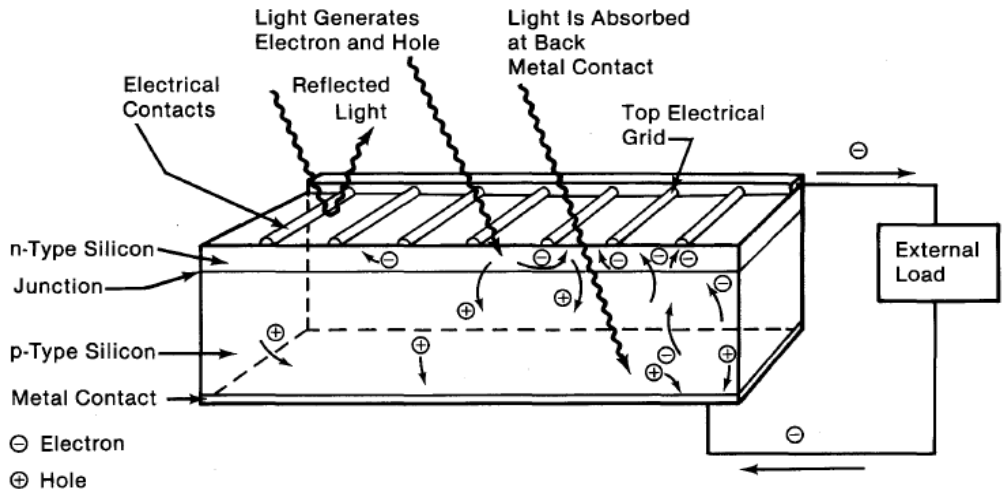

Figure 13 - Light incident on the cell creates electron-hole pairs, which are separated by the potential barrier, creating a voltage that drives a current through an external circuit. (Zweibel, 1982)

2.5.2. Grid-connected PV systems

PV cells are used to create PV modules that can then be used to create a PV array, which is the principal component of a grid-connected PV system.

Although small-scale PV systems may be applied in many different ways, the residential grid-connected PV systems are the most relevant configuration for this thesis scope. In these systems, any surplus of energy being produced is fed into the grid. Figure 14 and Figure 15 illustrate the functioning of the PV system and the required basic structures (PV array, Inverters, metering, controllers, and electrical devices) that allow an effective and safe interaction with the power grid. These are merely illustrative designs and variations are most likely possible.

Given the fact that a PV system generates electricity as DC and the one coming from the grid is alternating current (AC), an inverter is required to convert DC power from the PV array into AC power to be used by appliances on site or fed back into the grid via the meter. This conversion is possible due to the inverter's switching mechanism that allows the circuit to rapidly open and close. (Boxwell, 2013)

The grid-connected PV system uses grid-interactive inverters, also known as grid-tied inverters, which are crucial for the transfer of the electricity produced by a PV system into the grid. The grid-interactive inverter finds the maximum power available from the PV array to convert to AC and ensures that the power being fed into the grid is at the appropriate frequency and voltage. Most grid-interactive inverters include transformers that are used to increase the voltage to the level required by the grid.

When the grid is not operating within adequate voltage and frequency tolerances, the inverter has active and passive safety protections that allow shutting itself down. The inverter's ability to detect the grid's voltage and frequency is known as passive protection, whereas active protections is provided by the inverter detecting any frequency instability, frequency shift or power variation that would vary the voltage that the inverter detects. Moreover, the grid-connected inverter detects power cuts and monitors the power feed from the grid and if any extreme conditions occur it will disconnect, protecting not only the grid but also the PV system. Additionally, the amount of energy taken from the grid and fed back into the grid is monitored by the grid-connected meter. (Stapleton & Neill, 2012)

Other components involved in the PV system functioning are known collective as the balance of system (BoS) equipment and often must comply with local and/or national codes and regulations. These components are required to connect and protect the PV array and the inverter and includes cabling, disconnects/isolators, protection devices and monitoring equipment. (Stapleton & Neill, 2012)

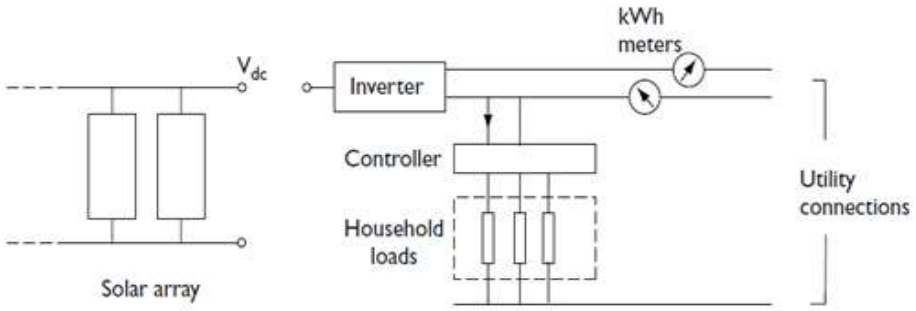

Figure 14 - Schematic of a grid-connected photovoltaic system. (Twidell & Weir, 2006)

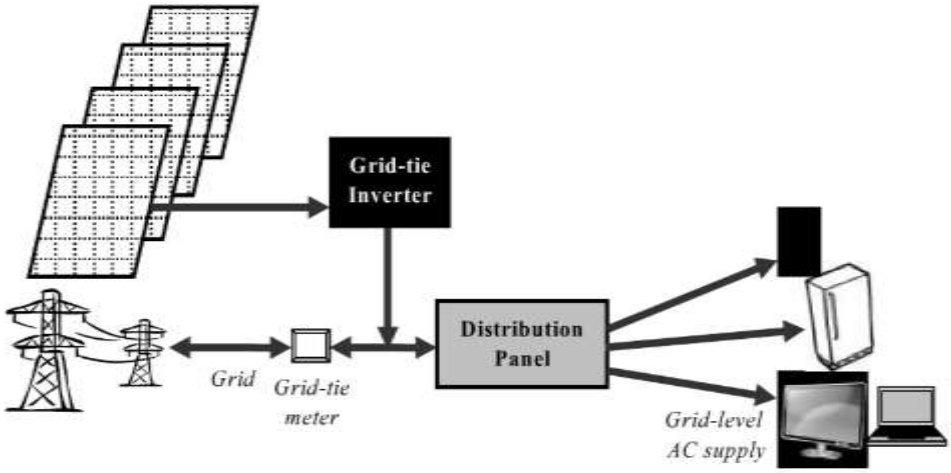

Figure 15 - Grid-connected system functioning. (Boxwell, 2013)

2.6. Time Series Forecasting

The present section introduces linear and nonlinear models that can be applied to typical time series collected from PV systems, in order to estimate future values of the energy production.

A time series is a set of numbers that measures the status of some activity over time. It is the historical record of some activity, with measurements taken at equally spaced intervals (exception: monthly) with consistency in the activity and the method of measurement.

2.6.1. Linear Models

Linear stochastic difference equation models with random input are often the statistical approach used to forecast time series. These stochastic models use past observations of the time series to predict future values. Such prediction can be used as a baseline to evaluate the possible importance of other variables to the systems.

The most significant of such models is the already introduced linear autoregressive integrated moving average (ARIMA) model. The basic idea behind these models is to find a mathematical equation that approximately generates the historical patterns. The ARIMA model is a type of self-projecting time series model that uses only the time series data of the activity to perform forecasts.

The Box-Jenkins Models are identified as AR (autoregressive), MA (moving average) and the combination of both is ARMA. AR models expresses a time series as a linear function of its past values and the order of the AR models tells the number of lagged past values included. MA models include lagged terms on the noise or residuals. Consequently, the ARMA model includes both types of lagged terms. The difference between ARMA and ARIMA is that that latter indicates that differencing was already applied to remove trends in the time series. *ARIMA (p, d, q)* defines models with an Autoregressive part of order p, a Moving average part order q and having applied d order differencing as illustrated in equation (26).

$$\left(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_\mathrm{p} B^p\right)(1 - B)^d \, y_t = c + \left(1 - \psi_1 B - \psi_2 B - \cdots \psi_\mathrm{q} B^q\right)\varepsilon_t \quad (26)$$

where φs and ψs are coefficients to be estimated, c is the constant level, B is the "backshit" operator and basically characterizes past values and *(1-B)* symbolizes the differencing operator. $(1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p)$ refers to the autoregressive polynomial and $\left(1 - \psi_1 B - \psi_2 B - \cdots \psi_q B^q\right)$ denotes the moving average polynomial.

To find the order of the operators *(p, q, d),* it is common practice to look at the autocorrelation function (ACF) and partial autocorrelation function (PACF), as well as the minimising information criterion - Akaike's information criterion (AIC) or Bayes information criterion (BIC) (Box, Jenkins, & Reinsel, 1994).

2.6.2. Nonlinear Models

The traditional approaches of the Box-Jenkins method assume that the time series in study are generated from linear processes. Though these linear processes may be advantageous to understand the details and are easier to explain and implement, several time series reveal unexplained features in a linear framework. Therefore, and in opposition to traditional statistical methods, nonlinear models such as artificial neural networks can capture those structures, are more flexible and have fewer limitations in estimating the essential relationships between the past values of the time series (inputs) and the future values (outputs).

2.7. Forecasting with Artificial Neural Networks

Artificial Neural Networks show powerful pattern recognition and pattern classification capabilities and have a wide range of applications, in science, business or industry. What makes artificial neural networks attractive is the fact that they possess self-adaptive methods that require few *a priori* model assumptions. Moreover, the relationships among the data are capture and they are capable of learning from examples and to generalize from experience, regardless of the complexity level. Usually, these models correctly infer the unobserved part of a population after the learning process, even if the sample data contains noisy information. Thus, forecasting is an ideal application area for artificial neural networks since it is performed through estimation of the future based on the past steps.

Artificial neural networks were firstly applied in 1964 by Michael Hu. However, the research was rather limited given the lack of training algorithms for the general model of multilayer networks at that time. In 1974, Paul Werbos formulated the concept of backpropagation and, in 1986; David Everett Rumelhart introduced the backpropagation algorithm, which fostered an enormous breakthrough for the development of artificial neural networks. Since these major findings, research efforts to further develop artificial neural networks have occurred and keep evolving until the present time.

These efforts focused in finding and developing an ideal model. This indicates selecting the most parsimonious model, that is, the model with the smallest number of parameters (Kriesel, 2005). This issue of finding a parsimonious model for a real problem became critical for all statistical methods and in particular to neural networks, given the possibility of overfitting. For example, in (Weigend, Hubernam, & Rumelhart, 1992), the authors addressed the problem of overfitting and proposed a method to overcome this network problem by introducing a term to the backpropagation cost function that penalizes network complexity.

At the same time, comparative studies between the performance of neural networks in forecasting and traditional statistical methods were developed. In (Zhang, Eddy Patuwo, & Hu, 1997) one can detect several inconsistent reports regarding the performance of artificial neural networks when used for forecasting. This is due to the fact that the network structures, sample data and training methods selected by the designer affects the network performance. For instance, linear data without much disturbance may explain that linear statistical models outperform artificial neural networks. Obviously, one cannot expect that an artificial neural network performs better than a linear model for linear relationships. Moreover, poor network design may also lead to poor performance. In (Tang & Fishwick, 1993), while comparing with traditional statistical methods, the authors found that artificial neural networks perform better for short memory series, when the forecast horizon increases and with more input nodes. Furthermore, using artificial neural networks, (Hill, Marquez, O'Connor, & Remus, 1994) achieved better results for monthly and quarterly time series forecasts than for yearly data. This is a consequence of the fact that monthly and quarterly time series data possess more irregularities and the artificial neural networks can detect the underlying pattern masked by noisy factors in a complex system.

In (Nelson, Hill, Remus, & O'Connor, 1994) the ability of an artificial neural network to learn seasonal patterns in a time series is discussed. The study indicated that forecasts could be more accurate if prior deseasonalization of seasonal time series is implemented. On the other hand, (Sharda & Patil, 1992) concluded that artificial neural networks are able to incorporate seasonality and its performance is not affected by seasonality of time series.

More recently, (Diaconescu, 2008) performs forecasting for different chaotic time series using a Nonlinear Autoregressive with exogenous inputs (NARX) dynamic recurrent neural network. The authors concluded that NARX recurrent networks can capture the dynamics of nonlinear dynamic systems and determined that the architecture of the tested model affects the performance of prediction. However, some drawbacks were also found such as the limitation in learning long time dependences. Additionally, (Menezes & Barreto, 2008) showed that a NARX network applied for long term multi-step-ahead predictions outperforms standard neural network based predictors, such as the time delay neural network (TDNN)[2] architectures.

Most researchers adopted, for specific problems, the trial-and-error methodology and, consequently, the literature can be rather inconsistent (Zhanget al., 1997). Though, as mentioned before, the artificial neural networks are promising alternatives to traditional statistical methods, they are a black-box method, which makes it difficult to explain the relationship between inputs and outputs. A wide range of questions still remain unanswered, but as any other method, artificial neural networks have their weaknesses and one should recognize them and generate the best possible suitable solution for its problem.

2.8. Solar Energy Forecasting with Artificial Neural Networks

The previous section introduced forecasting with artificial neural networks regardless of the field of study. However, this thesis is interested in assessing the forecasting potential of artificial neural networks applied to the solar forecasting domain, which is a more recent topic of study. This

[2] The TDNN model is similar to the NARX model but without the feedback loop.

section identifies relevant studies developed in the field of solar energy forecasting with artificial neural networks.

In (Sfetsos & Coonick, 2000) the authors used artificial neural networks to perform one-step ahead forecasting of hourly values of global irradiance and revealed that those results outperform linear models results. They also compared various models in terms of error and training time and found that the Levenberg-Marquardt algorithm achieved the best performance. Furthermore, the use an artificial neural network in (Mihalakakou, Santamouris, & Asimakopoulos, 2000) yielded a root mean square error RMSE 5% lower than the persistence approach.

Furthermore, the authors in (Cornaro, et al., 2009) compare models to forecast hourly solar irradiance with a day in advance. The models use artificial neural network techniques, where one of the models uses measured local data and the other is a hybrid model that also uses numerical weather prediction data. They conclude that the hybrid model gives the best results and improves almost 40% with respect to the persistence model.

Additionally, the authors in (Chaouachi, A.; Kamel, R.M.; Ichikawa, R.; Hayashi, H.; Nagasaka, K., 2009) studied the applicability of artificial neural networks for 1 day ahead solar power generation forecasting. Different types of networks were tested and a neural network ensemble is more precise than conventional networks (multi-layered perceptron, radial basis function, recurrent network), albeit all models demonstrate acceptable forecasting accuracy. Likewise, in (Yona, Senjyu, Saber, & Funabashi, 2007) a comparative study between different artificial neural networks models was conducted to predict insolation one day ahead, in which the Recurrent neural network outperforms the Feedforward neural network. Furthermore, the authors in (Paoli, Voyant, Muselli, & Nivet, 2010) presented a MLP neural network prediction approach to determine the global radiation at a daily horizon. They assumed an ad hoc time series preprocessing that reduces the error forecasts of about 5% compared to classical predictors. Additionally, in (Mantzari & Mantzaris, 2013) the researchers implemented a MLP neural network for half hour cloudiness forecasting and considered it an important tool for the estimation of cloudiness affecting solar radiation.

Artificial neural networks forecasting models for hourly solar irradiation for times of up to 6 days ahead were tested in (Marquez & Coimbra, 2011) and the authors concluded that the developed intelligent models outperformed satellite-based models. Moreover, an input selection scheme was used and results revealed that models with slightly larger sets of inputs generally perform better for same-day and 1-day ahead forecasts.

In (Di Piazza, Di Piazza, & Vitale, 2013) forecasting the daily solar radiation with two dynamic artificial neural networks (Feedforward Time Delay Neural Network and NARX) was proposed. According to the authors, both models had a satisfactory performance and can facilitate energy management of solar systems when storage systems are adopted.

All of the previous mentioned studies had an essential role in the development of this thesis. Though the parameters used in each study may be different, there were some important details to retrieve from the results, whether it was the input parameters, the applied algorithm, the type of neural network or even the time horizon for prediction.

Table 2 - Solar Forecasting - State of the art.

Author	Forecasting Time horizon	ANN	Error
Sfetsos and Coonick, (2000)	Hourly	Multilayer Perceptron	72% RMSE improvement comparing to the Persistence Model
Mihalakakou et al., (2000)	Hourly	Multilayer Perceptron	6% RMSE
Yona et al., (2007	Daily	Recurrent Network	15% MAPE
Chaouachi et al., (2009)	Daily	Recurrent Network	7% MAPE
Cornaro et al., (2009)	Daily	Hybrid model based on MLP	20% RMSE
Paoli et al., (2010)	Daily	Multilayer Perceptron	21% RMSE
Di Piazza et al., (2013)	Daily	NARX network	20% RMSE

2.9. Prediction Accuracy Evaluation

This section introduces the different existent tools to measure the overall accuracy of the network forecasts.

Forecasts are never completely accurate and will always deviate from the actual value. Consequently, the primary goal is to reduce as much as possible the associated error of the time series forecast. There is a wide range of functions that evaluate the neural network performance. However, these functions may not measure the *same units*, and therefore, it is not possible to compare the different functions between each other. Yet, the reason for presenting a wide range of forecast evaluators is the fact that each function accentuates specific features of the obtained values and observing all functions may help determine whether or not a neural network had a good performed.

To quantify the quality of a prediction, let us assume the time series target value $y(t)$ and the predicted value $y_f(t)$ for a series of length n. The difference between the sum of the squared deviations (SSE) of the forecasted value compared to the target value,

$$\mathrm{SSE} = \sum_{i=1}^{n} (y(t) - y_f(t))^2 \tag{27}$$

is highly dependent on the series size. Thus, a time series with more terms implicitly has a bigger error and a comparison of the quality of different time series forecasts is not feasible. To overcome this situation, one can use the mean value of this error:

$$MSE = \frac{1}{y_{max} - y_{min}} \frac{\sum_{i=1}^{n} \left(y(t) - y_f(t) \right)^2}{n} \tag{28}$$

where y_{max} and y_{min} are the maximum and minimum observed values.

Equation 28 calculates the mean squared error (MSE) of the predicted values. This is the first function to evaluate forecasts accuracy and measures the average of the square of the errors. This function incorporates both the variance of the prediction and its bias. MSE and variance have the same units of measurement as the square of the quantity being predicted. Consequently, one can take the square root of MSE, analogously to the standard deviation, to yield the root mean square error (RMSE). In fact, RMSE is basically the standard deviation of the differences between predicted values and target values. The RMSE is representative of the size of a "typical" error and will have the same units as the quantity of the time series being estimated. It tends to exaggerate large errors because squaring gives more weight to very large errors, which helps when comparing methods.

$$RMSE = \frac{1}{y_{max} - y_{min}} \sqrt{\frac{\sum_{i=1}^{n} \left(y(t) - y_f(t) \right)^2}{n}} \tag{29}$$

Another prediction evaluator commonly used is the mean absolute error (MAE) which is measured in the same units as the original data and is usually similar in magnitude to the root mean squared error, albeit slightly smaller. Taking the absolute value avoids the positives and negatives values canceling each other out.

$$MAE = \frac{1}{y_{max} - y_{min}} \frac{\sum_{i=1}^{n} \left| \left(y(t) - y_f(t) \right) \right|}{n} \tag{30}$$

Since RMSE error is more sensitive to occasional large errors, MAE may be the most relevant criteria when occasional large error is not a problem.

These functions solve the problem of cumulating error but still have the problem of relative error. The computed mean error is only absolute and highly depends on the series values. To overcome the relative error problem, one can calculate the coefficient of variation (CV), which is determined by the ratio between the standard deviation (σ) and the mean value of the evaluators.

$$\text{CV (\%)} = \frac{\sigma}{\overline{Evaluator}} \times 100 \tag{31}$$

The lower the CV value, the smaller the deviations between the multiple trials. Consequently, this may suggest a good model fit. Another important characteristic of the CV is the fact that is adimensional and allows comparisons between different models.

3. METHODOLOGY

This section intends to thoroughly describe and explain all steps for the design of a NARX model for solar forecasting. This model is implemented with MATLAB@ 2012^{Rb} Neural Networks Toolbox, which facilities the comprehension of the neural network functioning; however, ultimately, the user requires time to get acquainted with the neural networks mechanisms and some knowledge and experience handling datasets and computing functions.

3.1. Data Collection

Designing a neural network forecasting model requires several distinct steps because it involves the selection of many variables and parameters. A successful design can only be achieved if the problem is clearly specified and understood. Thus, in this particular work, the variables selection falls into those that are believed to be directly or indirectly influenced by cloud movements. Accordingly, the fundamental inputs selected were the time series of the ambient temperature and solar radiation, and data of five geographically separated PV systems from different households in the city of Utrecht, Netherlands, collected from the 1st of July to the 31st of July of 2012. The weather in the month of July (summer) in Utrecht usually remains pleasantly warm.

The meteorological data was collected from the Royal Netherlands Meteorological Institute (KNMI) website and the information regarding the PV systems was kindly provided by the PV-Group of the Copernicus Institute of Sustainable Development, Utrecht University. Figure 16 illustrates the geographic distribution of the PV systems (red circles) and the location where the Utrecht meteorological data is collected (small red rectangle). The PV systems will be designated as Centre, West, North, East, and South PV systems in order to differentiate them whenever required. Table 3 shows the maximum power installed and the date of installation of these PV systems.

Table 3 - PV systems technical information.

PV System	Max Power Installed (Watt)	Date of installation	Distance to Centre PV (km)
Centre	500	1 - Nov - 2003	-
West	560	29 - Nov - 2002	5
North	800	1 - Jan - 2004	7.5
East	1500	1 - Mar - 2002	3
South	500	1 - Nov - 2001	4

Figure 16 - Map of Utrecht illustrating the distribution of the PV systems.

3.2. Data Preprocessing

Data preprocessing is essential to analyze and transform the input and output variables to minimize noise, highlighting important relationships and flatten the distribution of variables so that the artificial neural network can learn relevant patterns. In fact, rarely the data collected for the input and output variables are fed into the artificial neural network in raw form.

The raw energy data collected from the PV systems inverters are in Watt-hour (Wh), with 1 min time steps; however, some of the time series have missing and/or outliers observations. The energy cumulative production is not as interesting to assess as the power production. This is due to the fact that, graphically, the power production shows the quick variations caused by cloud movements. Thus, data in energy units were converted to power units using the following equation.

$$y\,[Watt] = (\,x_i - x_{i-1}\,)\;kW.hour \times \frac{60\;\frac{min}{hour}}{1\;min} \times\; 1000\;\frac{W}{kW} \tag{32}$$

Differently, ambient temperature (in degrees Celsius) and solar radiation[3] (J/cm^2) data from the meteorological station have 1 hour time steps.

[3] In physics, J/cm^2 is typically designated as fluence (energy delivered per unit of area).

Missing observations can be handled in many different ways. In this work, all of those observations are considered as Not-a-Number (NaN) values, which is a function that considers undefined numbers and are ignored by the artificial neural network. Furthermore, outliers were removed by calculating a 10-points moving average (the average between the next five steps and the previous five steps of that observation).

Different time steps intervals were tested, in order to define which of them was able to capture the nonlinear and pattern characteristics of data. The time steps range assessed vary between 1min, 5min, 15min, 30min and 1h time steps. Though different time steps were tested, the selected forecast horizon was 1 day, which always coincided with predicting the last day of the month of July.

Manifestly, for simulation purposes both data from the PV systems and meteorological data must have the same length and, subsequently, the same time step value, when used simultaneously. Therefore, a moving average filter generates a series of averages of different subgroups of the full data set and later the time series length was shortened according to the time step required. In fact, this allows the training and testing sets to become more uniform distributed. On the other hand, the rapid changes caused by clouds on the solar power production are reduced and may disappear when the filter order severely increases.

Given the fact that the raw meteorological data has 1h time steps, the first phase was to interpolate the time series, using the MATLAB function *interp* to define time series with 1min steps.

Another preprocessing tool applied that may positively influence the artificial neural network performance was the removal of the observations between 10 pm and 6 am of each day. During these hours the PV systems cannot produce energy because it is night time and, consequently, the computational value is zero. A time series of the month of July with 1min time step has 44640 points of which 14880 points have the value zero (observations during the night). Accordingly, the difference between these values gives a total of 29760 points, which were used in the neural network. These "night values" add little to none information to the learning process and, thus, the complexity and simulations' running time were reduced by removing them.

To complete the data preprocessing, the time series were normalized between 0 and 1 because each dataset had different magnitudes. This process adjusts the measured values that have different scales and converts them to a common size; it is defined by the following equation:

$$x_{Normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{33}$$

where, x_{max} and x_{min} are the maximum and minimum points of the time series, respectively.

3.3. Training, Testing, and Validation sets

Before activating the network, the time series were divided into three different sets: training, testing and validation sets. Usually, the training set is larger because the patterns in the time series require learning. The testing set typically ranges in size from 10% to 30% of the training set and evaluates the generalization ability of a trained network. The final evaluation of the performance of the network was completed using the validation set. Frequently, the size of the validation set consists of the most recent observations in a way that there are enough remaining observations for both training and testing.

The range selected for the training, testing, and validation sets was 60%, 20%, and 20%, respectively, of the time series used as inputs. This division intends to avoid the risk of using a testing set characterized by a certain type of trend.

3.4. Artificial Neural Network Paradigms

There are countless ways to construct an artificial neural network. The properties of an individual neuron such as the transfer function and the way inputs are combined, associated with the number of neurons in each layer and the type of interconnections, define the neural network model. Consequently, the selection of the hidden layers, hidden layer neurons, and transfer function must be addressed in this section.

The generalization ability of an artificial neural network is provided by the hidden layer(s). Increasing the number of hidden layers also increases the possibility of *overfitting* and computation time, leading to poor results. The use of many hidden layers increases the number of weights relatively to the size of the training set and the ability to memorize instead of learning. Therefore, a single hidden layer was selected for the NARX network.

Moreover, selecting the number of hidden neurons involves a heurist approach. First, the default number of hidden neurons in the MATLAB@ 2012^{Rb} Neural Networks Toolbox, 10 hidden neurons, was tested to select the most adaptable time step (1min, 5min, 15min, 30min, and 1h). After the selection of the best time step, a range of hidden neurons (5, 10, 20, 35, and 50) was tested for each case that is proposed. With this experimentation, the NARX network with the best performance and, therefore, with the best ability to generalize, was selected and one can observe the impact of the hidden neurons in its performance.

The selection of the number of tapped delays is to a certain degree similar to the hidden neuron selection process. Initially, two tapped delays were used while selecting the hidden neurons number, which is also the default value in the MATLAB@ 2012^{Rb} Neural Networks Toolbox. Having the hidden neurons selected, the influence of feedback delays in the neural network was also tested. Similarly, a range of feedback delays was verified and the best performance was selected for different cases.

The transfer functions selected for the hidden layer and the output layer are the hyperbolic tangent and linear, respectively. These functions are also the default selection for time series prediction using MATLAB@ 2012^{Rb} Neural Networks Toolbox. Therefore, to be consistent with the transfer function being used, the input data was scaled between -1 and +1, according to the following equation,

$$x_{scaled} = y_{min} + (y_{max} - y_{min}) \times \frac{x - x_{min}}{x_{max} - x_{min}} \tag{34}$$

with y_{max} and y_{min} being +1 and -1 respectively. The data was scaled back to the original dimensions (0 to 1) after the network being processed.

3.5. Training

NARX network outputs are estimates of the outputs of the nonlinear dynamic systems. As illustrated in Figure 17, the output is fed back to the input of the Feedfoward neural network (Parallel architecture), which is part of the standard NARX architecture. However, the true output is available during the training process; therefore, a series-parallel architecture can be created.

Basically, the true output is used instead of feeding back the estimated output, providing static training and more accurate inputs to the Feedforward network.

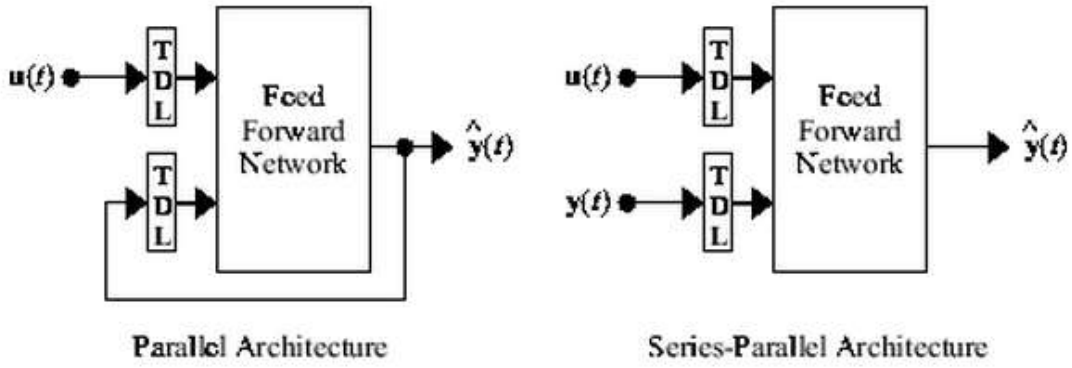

Figure 17 - NARX network architecture variations. (Beale et al., 2013)

The data from the Centre PV system is used as target series ($y(t)$) and the other variables are used as exogenous inputs ($u(t)$). Many different simulations were experimented to understand the impact that the considered exogenous variables have in the target data series.

In Figure 17, TDL designates Tapped Delay Line, which means, for instance, if two TDLs are used, the training begins with the third data point. In the series-parallel configuration, though there is no feedback TDL, as in the parallel architecture, $y(t)$ has a predetermined TDL value because $\hat{y}$(t) is a function of past values of $y(t)$ and the current/past values of $u(t)$.

The series-parallel configuration only presents errors for one-step-ahead predictions. Consequently, for multistep performances, the network has to be rearranged into the original parallel form. Though the information regarding the time of the month that we desire to predict is available, that data is only used to compare the final results.

Generally, the dynamic training (iterated) of the parallel architecture takes longer and the performance is not as good as that obtained with series-parallel training. Therefore, the NARX network is trained with the series-parallel configuration.

Every time a network is trained a different solution is achieved given the different initial weights and bias values, i.e., different outputs may be achieved with the same inputs. Thus, to ensure good accuracy, each specific architecture was simulated eleven times and the median of the eleven simulations was calculated; results larger than 15% of the median are disregarded, to eliminate outliers. Furthermore, the mean value of the remaining results is calculated and assumed as the final value of the performance of that specific network architecture.

In this work, the Lavenberg-Marquardt algorithm is the algorithm used for every training process and the number of epochs, which corresponds to the number of iterations through all the series that the network will perform in the training process, is set to a maximum of 1000 (default value of MATLAB@ 2012^{Rb} Neural Networks Toolbox). Moreover, the number of training interactions is defined automatically as the early stopping principle is applied, i.e. the training stops when the improvement of the error function is no longer possible.

3.6. Evaluation

The evaluation module intends to compare and understand the level of accuracy of forecasts according to the indicators that have been presented in the "prediction accuracy evaluation" section. Thus, one can compare different architectures and adjust the parameters in order to obtain the best possible prediction.

3.7. Optimization

The following different scenarios are applied to help selecting the best network architecture and to learn about the impact of using variations of the exogenous variables.

Case 1 - Selection of the time step

This case applied different time steps (1min, 5min, 15min, 30min, and 1h) using 4 exogenous inputs, 1 output, and the default values of 10 hidden neurons and 2 tapped delay lines. The goal was to determine the most appropriate time step for the NARX network. The architecture of this example can be hence described as 4 – 10 – 1 with 2 TDL NARX network.

Table 4 - Case 1 configuration.

Inputs	4 - Present and past values of the (Exogenous) time series of the West, North, East, and South PV systems. 1 - Past values of the time series of the Centre PV system
Output	1 - Future values of the time series of the Centre PV system
Number of hidden neurons	10
Number of Tapped Delay Line	2
Time step	***Variable (1min, 5min, 15min, 30min, and 1h)***

Case 2 - Selection of the best configuration of a NARX network with data of 4 PV systems as exogenous inputs.

After the ideal time step selection, the influence of the hidden neurons and tapped delay line was tested. The number of hidden neurons tested varied between 5, 10, 20, 35 and 50 using a default value of a 2 tapped delay line.

Additionally, after the number of hidden numbers selection, several tapped delay lines were tested and the network that featured the lowest errors was selected. The tapped delay line experimented varied according to the time step selection of the case 1.

This case aimed to assess if the information of other PV systems, surrounding the system of interest, is relevant for the forecast accuracy when compared to different cases.

Table 5 - Case 2 configuration.

Inputs	4 - Present and past values of the (Exogenous) time series of the West, North, East, and South PV systems. 1 - Past values of the time series of the Centre PV System.
Output	1 - Future values of the time series of the Centre PV system

Case 3 - Selection of the best configuration of a NARX network with data of 2 PV systems as exogenous inputs.

The network configuration was defined based on the same methodology of the former case. After establishing the configuration, the variations of parameters were now focused on the inputs, in order to determine if removing data from the NARX network actually affected the final result. In this case, combinations of data of the West and East PV systems and data of the North and South PV systems were used as exogenous inputs. That is, 2 different simulations were performed.

Table 6 - Case 3 configuration.

Inputs	2 - Present and past values of the (Exogenous) time series of the West and East PV systems **and** North and South PV systems. 1 - Past values of the time series of the Centre PV System.
Output	1 - Future values of the time series of the PV Centre system.

Case 4 - NARX network with meteorological data as exogenous inputs

The present case intended to determine the NARX network performance using solely meteorological data (radiation and temperature) as exogenous inputs.

Table 7 - Case 4 configuration.

Inputs	2 - Present and past values of the (Exogenous) time series of the Solar Radiation and Temperature. 1 - Past values of the time series of the Centre System.
Output	1 - Future values of the time series of the Centre system.

Case 5 - NARX network with data of 4 PV systems and meteorological data as exogenous inputs (6 exogenous inputs total)

The final model used all the available data and intended to determine whether or not the forecasts can be improved by adding positively correlated information to the network.

Table 8 - Case 5 configuration.

Inputs	4 - Present and past values of the (Exogenous) time series of the West, North, East, and South PV systems. 1 - Past values of the time series of the Centre PV System. 2 - Present and past values of the (Exogenous) time series of Temperature and Radiation.
Output	1 - Future values time series of the Centre PV system.

Case 6 - Multistep ahead forecasting

The different scenarios which were previously presented performed 1 day predictions on a one step forecast basis. To be precise, to forecast the PV output at $t+1$ we use the input time series until t. This allowed the determination of the most relevant data and selection of the best configuration amongst the previous scenarios.

On the other hand, the present case intends to study the multistep ahead NARX neural network forecasting performances, which means, trying to forecast the PV output at $t+1$ a few hours in advance. Several time steps (5min, 30min, 1h, 1h30, 2h, 2h30, 3h, 4h, 6h, 8h, 12h and 16h) are executed and tested for a horizon of up to 1 day in advance.

4. RESULTS

4.1. Raw and Preprocessing data

Figure 18 and Figure 19 illustrate the PV systems time series converted to power (Watt) and the meteorological time series raw data, respectively. One can verify that the East and South PV systems time series present several missing observations in the first days of the month. Moreover, only 5 days of the month display the shape of the typical "clear sky day", which indicates that most days in Utrecht are highly affected by cloud formations. The variation on the radiation time series is obviously very consistent with the shape of the PV system time series. Also, it is clearly observable that these meteorological data are positively correlated with the PV system time series because the temperature increases[4] when a sunny day is detected and the opposite is also verified.

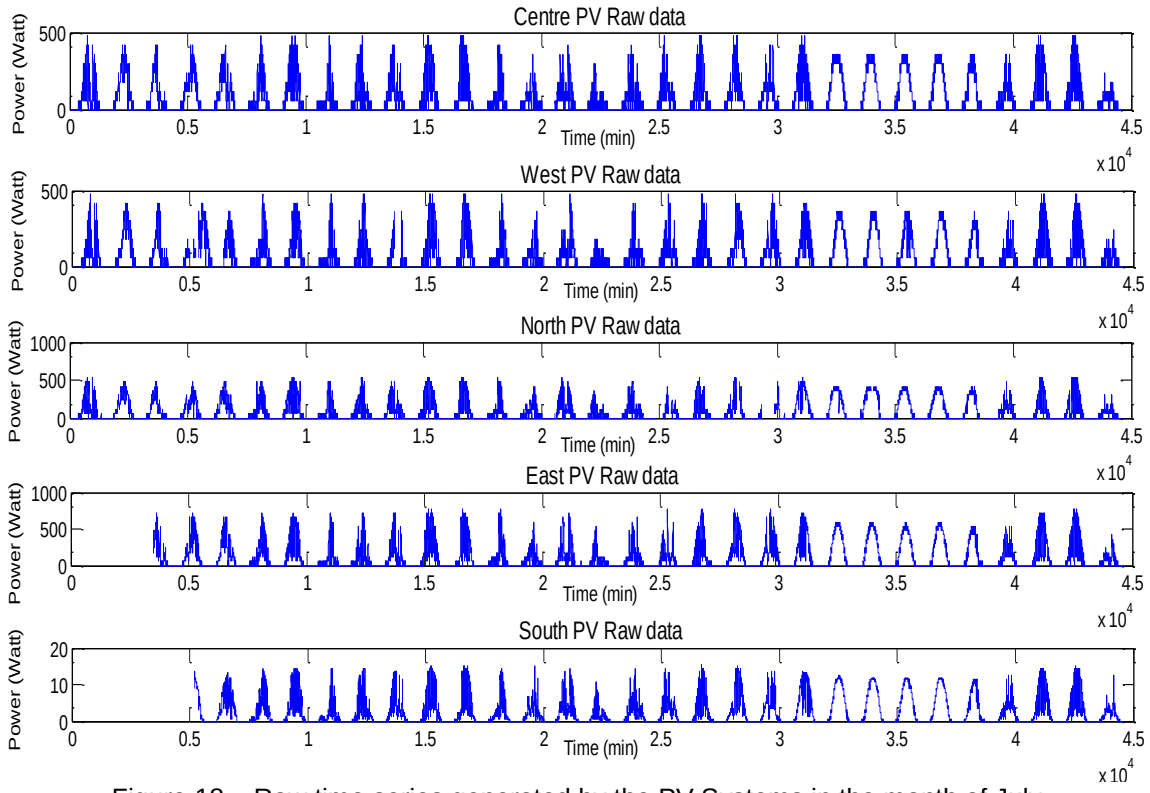

Figure 18 – Raw time series generated by the PV Systems in the month of July.

[4] There is a direct correlation between the ambient temperature and the solar radiation. Although the PV cell efficiency decreases with the temperature, the solar radiation effect is more preponderant. Therefore, one may say there is a correlation between the PV production and the ambient temperature.

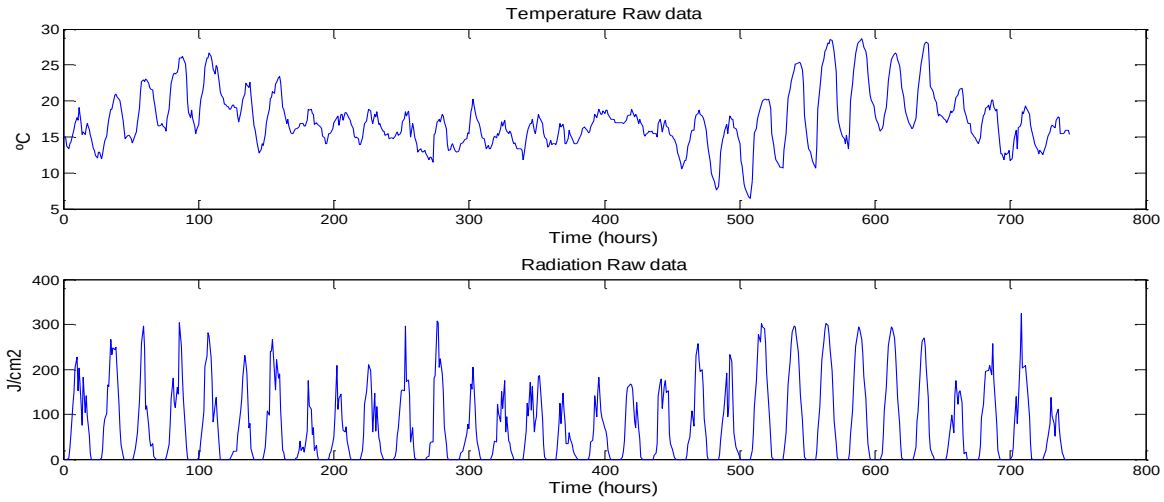

Figure 19 – Meteorological raw data in the month of July.

Furthermore, the following two figures illustrate the results of the preprocessing of the time series of the Centre PV system and the meteorological data. Although the other PV systems time series are not presented, the same processing was applied to them. Figure 20 displays normalized moving average time series with different time steps.

Figure 21 illustrates the interpolated meteorological time series with a 5 min step. As before, interpolated meteorological time series with 1min, 15min, and 30min time steps are not presented but have undergone the same processing.

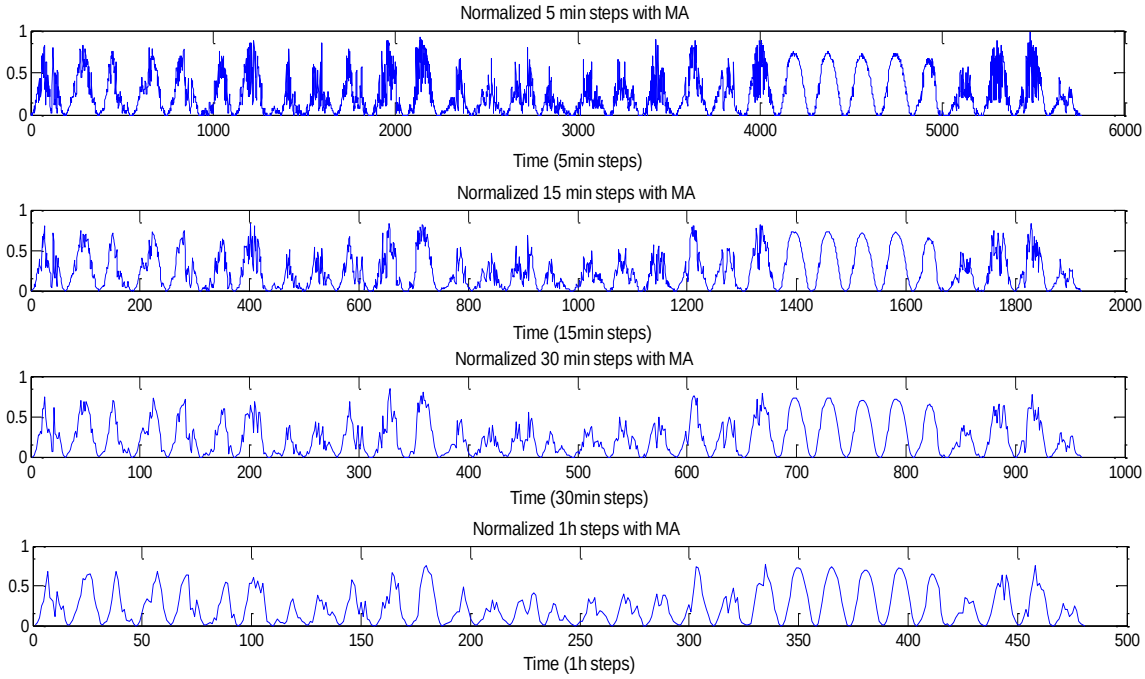

Figure 20 - Preprocessing of time series data generated by the Centre PV system.

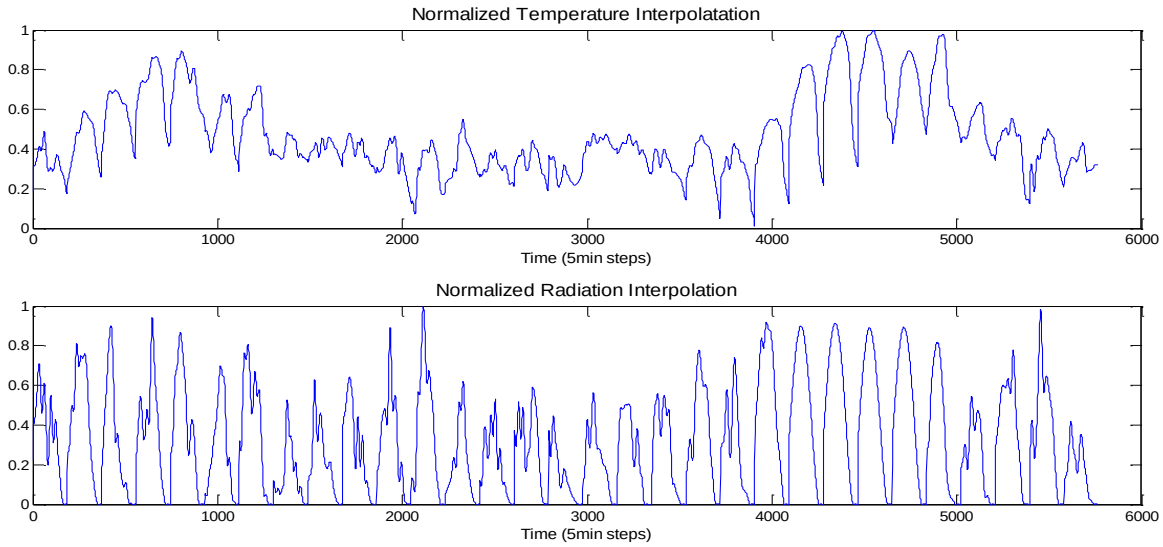

Figure 21 - Normalization and interpolation of the meteorological time series.

4.2. Case 1 - Selection of the time step

Table 9 and Table 10 show the results of a 4-10-1 with 2 TDL NARX network. For illustration purposes "t" and "f" characterize the performances of the series-parallel configuration and the parallel configuration, respectively. "t" denotes the training and modelling of the NARX network, whereas "f" denotes the multistep predictions/forecasting process. Moreover, the tables are coloured to illustrate the best (green), intermediate (yellow), and worst (red) relative performances.

Table 9 - Case 1Training and predicting error results using different time steps.

Time step	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
1min	0.017	0.093	0.064	0.007	0.061	0.046
5min	0.014	0.084	0.051	0.003	0.038	0.027
15min	0.014	0.083	0.054	0.005	0.050	0.036
30min	0.014	0.086	0.062	0.012	0.081	0.065
1h	0.016	0.093	0.065	0.010	0.069	0.056

Table 10 - Case 1 Training and predicting coefficients of variation, using different time steps.

Time step	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
1min	2%	1%	2%	4%	4%	4%
5min	3%	2%	4%	6%	5%	11%
15min	5%	2%	4%	10%	11%	7%
30min	8%	6%	7%	42%	21%	23%
1h	12%	9%	8%	29%	19%	15%

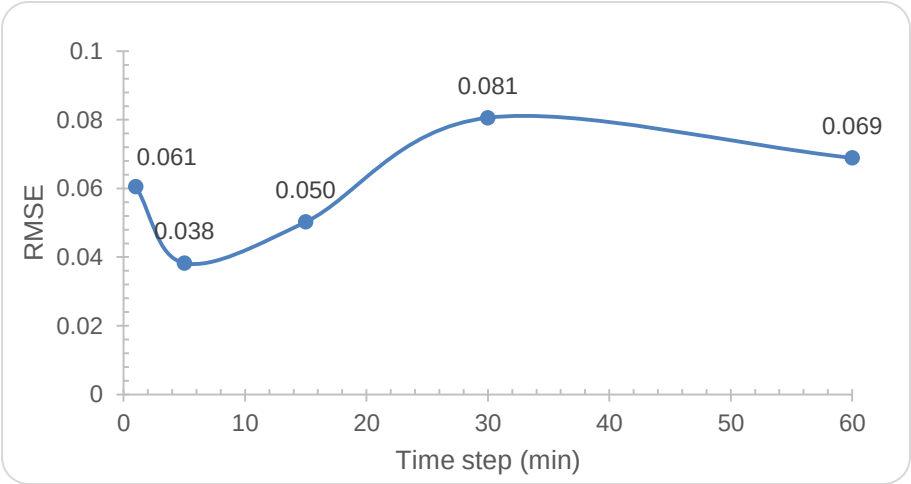

Figure 22 – Case 1 RMSE results for the forecasting process using different time steps.

RMSE is one of the most important evaluators for the assessment of the quality of the network and its values are always higher than MAE. Therefore, though these evaluators are not comparable between them, both might exhibit the same behaviour when comparing different simulations.

Figure 22 displays the NARX networks RMSEf results using different time steps. These results led us to select the 5 min time step as the most appropriate to perform all remainder cases. The network forecasting performance of 30 min and 1h time step were clearly the worst, albeit still showed an acceptable error, as displayed in Table 9 and Table 10. In spite of mean errors being acceptable, there was a significant variation of the independent performances. The time series of 30min and 1h time steps probably did not have enough length to allow the network to capture enough patterns and complete a good network learning process. That is, given the fact that only the month of July is being assessed, 30min and 1h time steps for a single month results in a too small length time series.

Though the performance of the 1min time step network was rather good, especially because of the low values of the coefficient of variation (around 1% for the training and 3% to 4% for the forecasting), it takes a very long time to process because of the high number of weights involved in such complex network.

Finally, despite of the RMSEf values of the network with the time series with 15 min time step showing good accuracy, it did not outperform the network with the time series with 5 min time step. Though the coefficient of variation of both performances was similar, the 5 min time step network demonstrates better forecasting results and, therefore, it is more capable of learning the patterns of the time series. This may be a consequence of the fact that many physical phenomena that establish the correlation between the inputs and the output can be detected in a 5min timeline.

4.3. Case 2 - NARX network with data of 4 PV systems as exogenous inputs.

In case 1, the number of hidden neurons and tapped delay line were selected based on the default values of MATLAB@ 2012^{Rb} Neural Networks Toolbox. However, for case 2, though the neural network has the same exogenous inputs as in case 1, the 5 min time step is now used and several simulations were made to determine the effect of the hidden neurons and the tapped delay

line in the network performance. First, the tapped delay line was maintained constant using the default value (2 TDLs) while the number of hidden neurons varied. Subsequently, the number of hidden neurons that yielded the best neural network performance was selected and maintained constant while a range of tapped delay lines were tested.

Selection of the number of hidden neurons

Table 11 – Case 2 training and predicting error results, using different hidden neurons.

Hidden Neurons	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5	0.014	0.083	0.050	0.003	0.041	0.029
10	0.014	0.084	0.051	0.003	0.038	0.027
20	0.014	0.083	0.051	0.003	0.042	0.029
35	0.014	0.084	0.053	0.003	0.041	0.029
50	0.015	0.088	0.054	0.006	0.056	0.043

Table 12 – Case 2 training and predicting coefficients of variation, using different hidden neurons.

Hidden Neurons	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5	3%	2%	3%	8%	7%	10%
10	3%	2%	4%	6%	5%	11%
20	3%	1%	3%	10%	5%	8%
35	6%	3%	6%	21%	11%	12%
50	9%	6%	6%	39%	21%	27%

Table 11 and Table 12 show the results obtained for the NARX network varying the number of hidden neurons in the hidden layer. First, it is observable that the parallel configuration (multistep predictions) results are relatively smaller than the results from the series-parallel configuration (training). Therefore, one can verify that a recurrent network such as the one that is used to perform multistep predictions outperforms the multilayer Feedforward network, which is used for the training process (series-parallel configuration). However, looking at the Table 12, one confirms that the parallel configuration results are considerably higher compared to the results from the series-parallel configuration (training). Indeed, though the multistep forecasting process shows good performances, the coefficients of variation lead us to consider that a high level of variation of performances may occur. On the other hand, the performance of the training process shows that fewer variations occur, albeit the errors slightly increase. The low errors of the multistep prediction process result from the fact that the neural network was able to learn adequately and to generalize.

Though the functions that measure the errors magnitude cannot be compared between each other, one may observe that the results are coherent. That is, for instance, when the RMSEf increases the MAEf also tends to increase. In Table 11, training and forecasting errors tend to increase with the number of hidden neurons. The RMSE penalizes higher deviations and it is the most interesting evaluator because we intend that the predictions do not diverge significantly from the true target value.

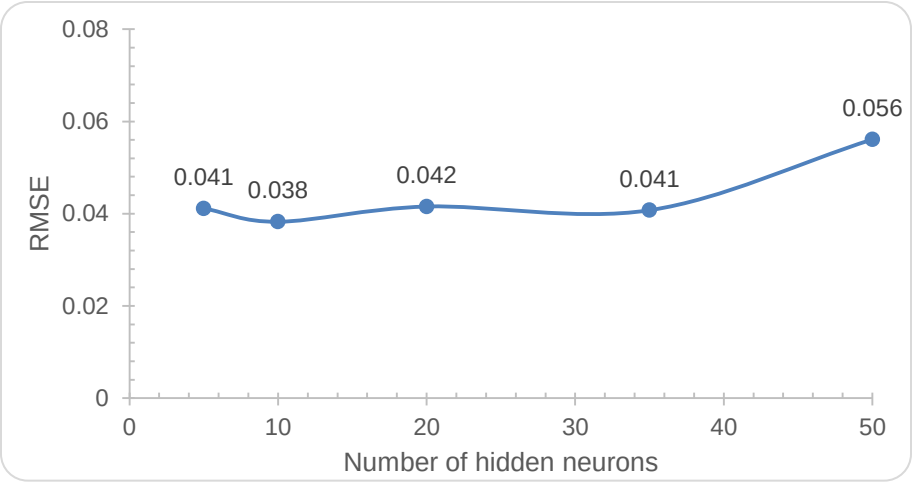

Figure 23 - Case 2 RMSE results for the forecasting process using different hidden neurons.

Figure 23 illustrates, graphically, those variations of the RMSEf function. The difference between the value of the best performance and the worst performance of the RMSEf is considerably large (more than 50% relative difference). We want to select the number of hidden neurons that implies less number of weights and complexity of the network but still has a good performance, that is, the most parsimonious model.

Accordingly, one observes that the neural network's lowest RMSEf of 0.038 was achieved with 10 hidden neurons. The network with 5 hidden neurons produced a slightly higher error value because it was not as effective in capturing the underlying features of data. Moreover, the remainder number of hidden neurons in the network configuration also did not perform as well as the 10 hidden neurons configuration given the fact that adds unnecessary weights.

Selection of the tapped delay line

Table 13 and Table 14 show the neural network results using different tapped delay lines, and once again the colors allow us to easily identify the best and the worst performances. The associated real time that is being considered in every tapped delay line is also shown.

Table 13 - Case 2 training and predicting error results, using different TDL.

Real Time	TDL	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
10min	2	0.014	0.084	0.051	0.003	0.038	0.027
20min	4	0.014	0.083	0.050	0.005	0.050	0.036
30min	6	0.013	0.082	0.050	0.003	0.040	0.028
45min	9	0.014	0.083	0.051	0.004	0.044	0.033
1h	12	0.014	0.083	0.051	0.004	0.047	0.034
1h30	16	0.014	0.085	0.054	0.005	0.051	0.040
2h	24	0.013	0.083	0.051	0.005	0.050	0.037
3h	36	0.015	0.089	0.056	0.007	0.063	0.049
4h	48	0.020	0.104	0.067	0.010	0.070	0.059

Table 14 - Case 2 training and predicting coefficients of variation, using different TDL.

Real Time	TDL	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
10min	2	3%	2%	4%	6%	5%	11%
20min	4	6%	3%	3%	24%	17%	13%
30min	6	3%	1%	4%	10%	5%	10%
45min	9	5%	2%	5%	23%	16%	12%
1h	12	4%	3%	4%	20%	10%	16%
1h30	16	7%	4%	8%	27%	13%	16%
2h	24	3%	5%	5%	31%	18%	18%
3h	36	8%	7%	8%	37%	22%	22%
4h	48	15%	10%	13%	32%	17%	28%

It was observed that adding more numbers to the tapped delay line increases the complexity of the network and, consequently, the error results. A considerable amount of time was required to compute the networks with a 48 Tapped delay line and understandably the overall performance of that network configuration is significantly poor.

One may observe that the training results of the neural networks appear to be very consistent, with small variations. This suggests similar accuracy and that varying the tapped delay line only poses relevant discrepancies in the training results when it is severely increased. On the other hand higher variations can be detected for the multistep predictions in the coefficient of variation's table, albeit those variations are acceptable considering the task. Usually, for multistep predictions, the best neural network performances exhibit a lower coefficient of variation and, on the other hand, worse performances tend to have higher coefficients of variation.

All functions suggest similar trends for the errors. Figure 24 demonstrates that the performance of the NARX network tends to deteriorate by increasing dramatically the TDL.

The configuration using 2 TDL (10 min) revealed to be the most parsimonious model with an RMSEf of 0.038. The difference between the latter result and the one with the poorer performance was higher than 80%. Accordingly, one can determine that for a NARX neural network with 4 PV systems data as exogenous inputs, it is more relevant to feedback to the network only the information of the two previous outputs than feedback several of the previous output values. This is coherent with most real world dynamic systems, where future values tend to be similar to the closest previous values.

Furthermore, it was also observed that the network with 6 TDL and the best configured network (2TDL) had very similar results. Therefore, one can verify that periodically adding information may improve the network performance. In this case, this extra past observations considered may indicate that cloud movements were detected.

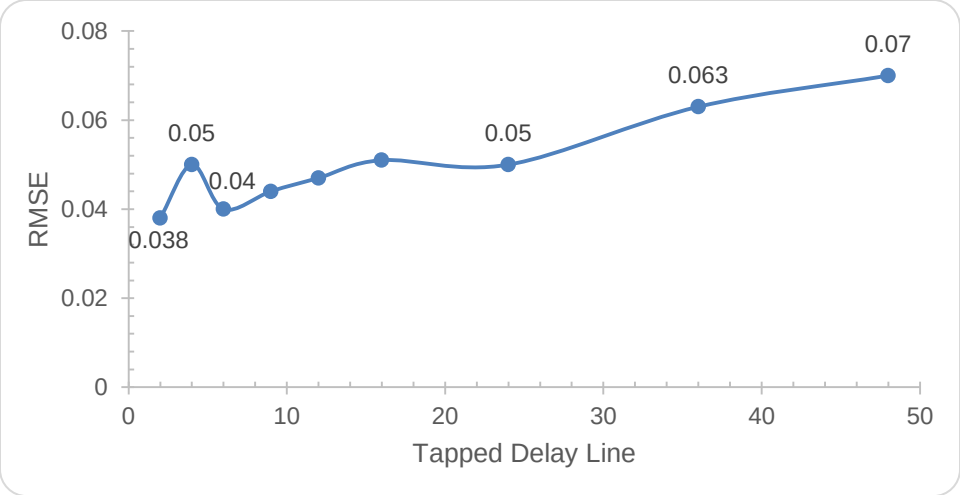

Figure 24 - Case 2 RMSE results for the forecasting process using different TDL.

4.4. Case 3 - NARX network with data of 2 PV systems as exogenous inputs

To compare models and assess the influence of the exogenous inputs in the NARX neural network, different combinations of inputs were applied. First, the neural network was tested using solely the information of the West and East PV systems as exogenous inputs, and the second test uses the data of the North and South PV systems as exogenous inputs. As standard practice, the ideal number of hidden neurons and tapped delay line were determined using the same method applied in the previous case.

Case 3.1 West and East PV systems as exogenous inputs

Selection of the number of hidden neurons

Table 15 and Table 16 exhibit the results of different neural network configurations using data from the West and East PV systems as exogenous inputs. The MSEt, RMSEt and MAEt in Table 15 indicate very consistent and accurate performances of the series-parallel network configuration, that is, the training procedure. According to the MSEf, RMSEf and MAEf results in Table 15, the network with 10 hidden neurons outperforms the other network configurations for multistep predictions. In Figure 25 one can verify that, similarly to previous cases, increasing the number of hidden neurons also increases the final error, by a slight difference. Consequently, 10 hidden neurons are selected for the neural network configuration. Moreover, though in Table 16 the configuration with 10 hidden neurons did not show the lowest coefficients of variation results, the observed variations were satisfactory.

Table 15 - Case 3.1 training and predicting error results, using different hidden neurons.

Hidden Neurons	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5	0.013	0.082	0.050	0.004	0.045	0.032
10	0.014	0.082	0.050	0.004	0.043	0.031
20	0.014	0.082	0.051	0.004	0.045	0.034
35	0.014	0.083	0.051	0.005	0.050	0.038
50	0.014	0.084	0.051	0.005	0.052	0.040

Table 16 - Case 3.1 training and predicting coefficients of variation, using different hidden neurons.

Hidden Neurons	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5	2%	1%	3%	12%	8%	9%
10	4%	2%	5%	19%	10%	16%
20	4%	2%	3%	26%	14%	22%
35	5%	3%	5%	21%	13%	15%
50	6%	5%	6%	36%	20%	29%

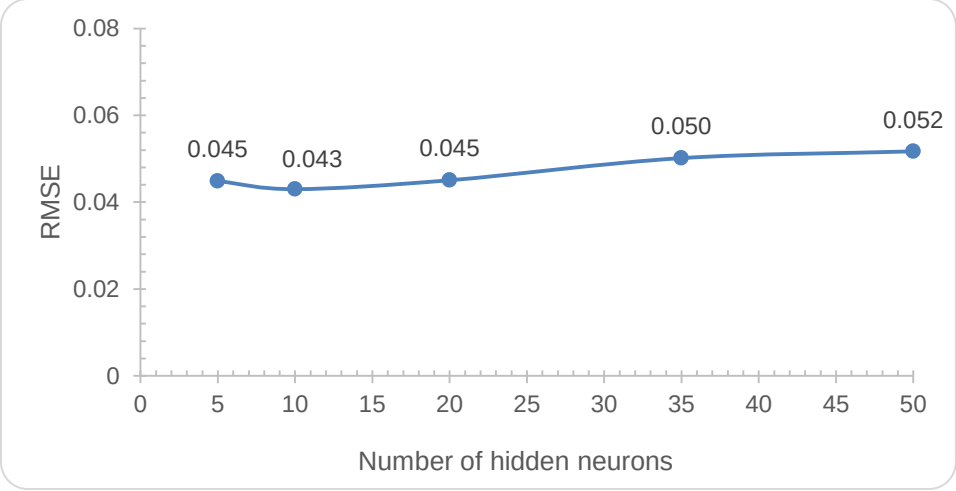

Figure 25 - Case 3.1 RMSE results for the forecasting process using different hidden neurons.

To achieve the best network configuration, a range of tapped delay lines were tested using the selected number of hidden neurons (10 hidden neurons).

Selection of the tapped delay line

One might have noticed consistent error values between different configurations and low variability of results for the neural network training process. On the other hand, though the recurrent network used for multistep predictions yields lower MSE, RMSE and MAE results, considerably higher coefficients of variation were observed. The ideal number of tapped delay line is selected based on the lowest error of the multistep predictions procedure. Accordingly, a neural network with a 2 TDL configuration showed to be the best fit, with the RMSEf and the MAEf values of 0.043 and 0.031 respectively. The latter configuration also showed low coefficients of variation.

Table 17 - Case 3.1 training and predicting error results, using different TDL.

Real Time	TDL	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
10min	2	0.014	0.082	0.050	0.004	0.043	0.031
20min	4	0.013	0.081	0.048	0.004	0.046	0.034
30min	6	0.013	0.081	0.049	0.005	0.053	0.042
45min	9	0.013	0.082	0.049	0.008	0.065	0.048
1h	12	0.013	0.081	0.049	0.006	0.056	0.042
1h30	16	0.013	0.080	0.049	0.008	0.065	0.050
2h	24	0.013	0.082	0.051	0.006	0.060	0.045
3h	36	0.014	0.083	0.052	0.006	0.057	0.044
4h	48	0.013	0.082	0.052	0.005	0.052	0.038

Table 18 - Case 3.1 training and predicting coefficients of variation, using different TDL.

Real Time	TDL	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
10min	2	2%	1%	2%	11%	6%	6%
20min	4	3%	1%	3%	6%	5%	11%
30min	6	3%	3%	3%	16%	10%	13%
45min	9	3%	3%	4%	48%	25%	28%
1h	12	4%	2%	3%	30%	19%	21%
1h30	16	5%	3%	3%	13%	7%	10%
2h	24	5%	3%	5%	18%	14%	15%
3h	36	7%	4%	5%	20%	14%	17%
4h	48	4%	2%	4%	14%	9%	12%

One can also observe, graphically, in Figure 26 Figure 26, the RMSEf values of the different simulations. We can verify that the performance of the neural network did not improve by adding more TDL. In fact, considering past values up to a 9 TDL (45 min) did not prove to be relevant, as the error kept increasing. However, between the simulation with the neural network with the 9 TDL (45 h) configuration and the 12 TDL (1 h) configuration, a sudden reduction of around 15% was detected, which indicates that the extra 15 min considered may involve fundamental information to perform the predictions. Furthermore, the accumulated values of 1h30

used as tapped delay line also did not pose relevant variation in the RMSEf. Yet, beyond the 16 TDL, we can identify that the RMSEf values tend to decrease but never reach or outperform the minimum RMSE achieved with a neural network with a 2 TDL configuration.

Thus, the best NARX neural network configuration using data from the West and East PV systems as exogenous inputs has 10 hidden neurons, 2TDL and a 0.043 RMSEf and a 0.031 MAEf.

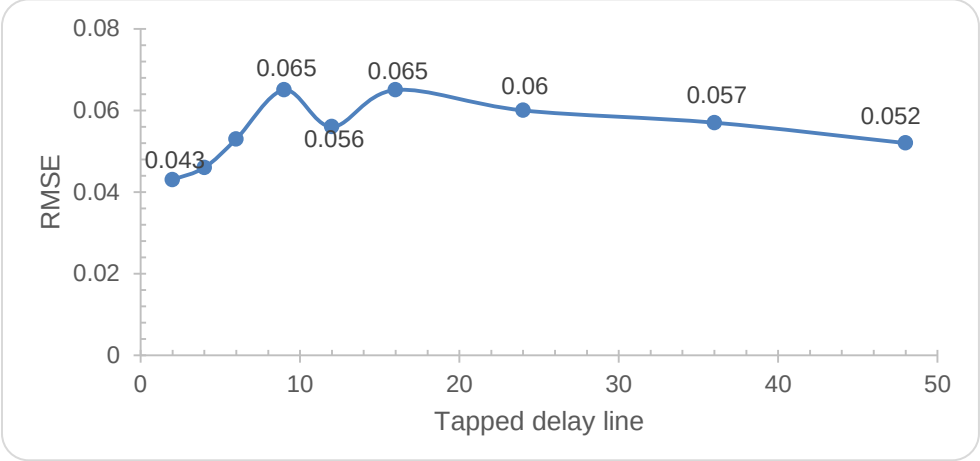

Figure 26 - Case 3.1 RMSE results for the forecasting process using different TDL.

Case 3.2 North and South PV systems as exogenous inputs

Selection of the number of hidden neurons

In Table 19 and Table 20, one can compare the results of different neural networks configurations using time series data of the North and South PV system as exogenous inputs. The training process showed close similarities between the different configurations, which is consistent with all the previous cases. However, as displayed in Table 20, unlike the previous cases, the neural network with 5 hidden neurons showed the lowest MSEf, RMSEf and MAEf results. One can also observe in Figure 27 that the network configuration with 20 hidden neurons and 35 hidden neurons had the 0.059 RMSEf, which is also the same result as the network configuration with 5 hidden neurons. However, the MAEf function indicates better results with the 5 hidden neurons network configuration. Accordingly, the latter configuration is selected as the most parsimonious model.

Table 19 - Case 3.2 training and predicting error results, using different hidden neurons.

Hidden Neurons	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5	0.017	0.091	0.055	0.007	0.059	0.043
10	0.017	0.091	0.055	0.008	0.066	0.046
20	0.017	0.092	0.056	0.007	0.059	0.045
35	0.017	0.091	0.055	0.007	0.059	0.044
50	0.017	0.093	0.056	0.018	0.094	0.073

Table 20 - Case 3.2 training and predicting coefficients of variation, using different hidden neurons.

Hidden Neurons	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5	1%	1%	2%	9%	5%	7%
10	2%	1%	2%	26%	16%	17%
20	3%	3%	3%	21%	11%	16%
35	3%	2%	5%	20%	11%	13%
50	5%	3%	5%	47%	25%	32%

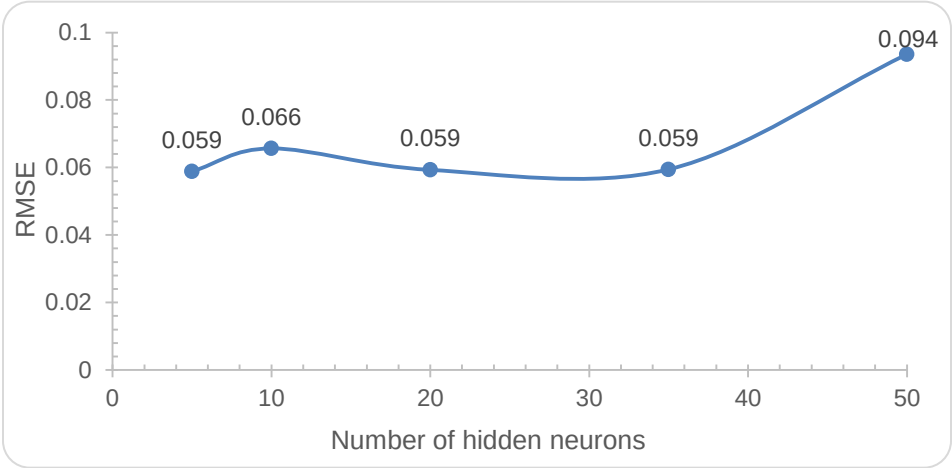

Figure 27 - Case 3.2 RMSE results for the forecasting process using different hidden neurons.

Selection of the tapped delay line

To complete the configuration selection of this case, the following Table 21, Table 22 and Figure 28 introduce the network results using different tapped delay line.

Table 21 - Case 3.2 training and predicting error results, using different TDL.

Real Time	TDL	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
10min	2	0.017	0.091	0.055	0.007	0.059	0.043
20min	4	0.016	0.089	0.053	0.006	0.055	0.041
30min	6	0.016	0.089	0.054	0.007	0.061	0.044
45min	9	0.016	0.089	0.054	0.006	0.056	0.041
1h	12	0.016	0.091	0.056	0.007	0.059	0.044
1h30	16	0.017	0.092	0.055	0.010	0.071	0.055
2h	24	0.017	0.091	0.056	0.009	0.065	0.047
3h	36	0.017	0.093	0.059	0.017	0.091	0.071
4h	48	0.018	0.094	0.059	0.013	0.080	0.060

Table 22- Case 3.2 training and predicting coefficients of variation, using different TDL.

Real Time	TDL	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
10min	2	1%	1%	2%	9%	5%	7%
20min	4	1%	1%	2%	20%	11%	18%
30min	6	3%	1%	4%	24%	14%	16%
45min	9	2%	1%	2%	12%	8%	11%
1h	12	3%	1%	5%	20%	10%	12%
1h30	16	5%	4%	3%	43%	23%	29%
2h	24	4%	2%	4%	30%	15%	17%
3h	36	3%	3%	6%	40%	20%	22%
4h	48	5%	3%	7%	22%	11%	12%

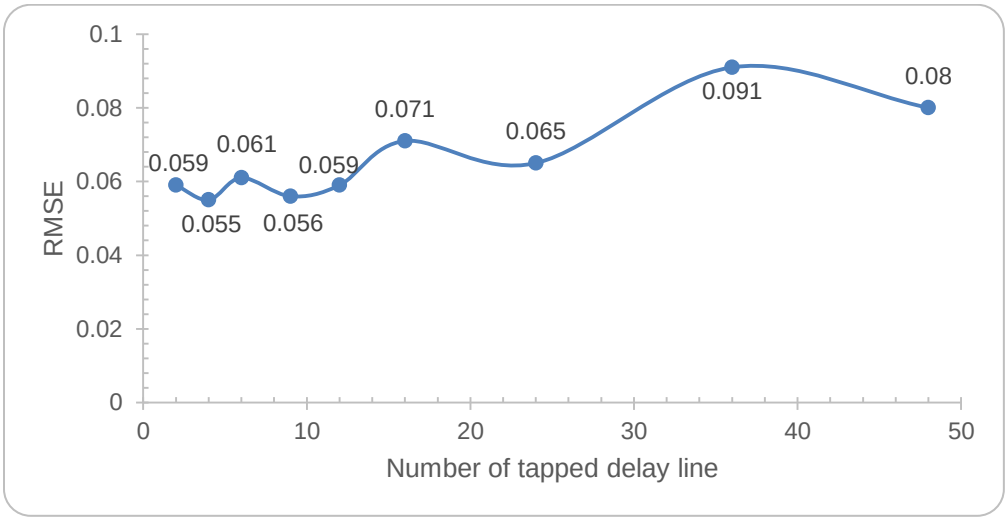

Figure 28 - Case 3.2 RMSE results for the forecasting process using different TDL.

According to the latter figure and tables, a 4 TDL was selected as the configuration that yielded the lowest errors (0.055 RMSEf). Figure 28 shows periodical oscillations, which indicates that using periodically past information of the North and South PV systems as exogenous inputs may have added relevant value to the network. Besides the global minimum at the 4 (20 min) TDL configuration, two local minimums were achieved with the 9 TDL (45min) and 24 TDL (2h) network configurations. This reveals that, having the 4 TDL as reference, using another 25 min of information and another 1h10 min also improves the performance, that is, possibly cloud movement information was responsible for the error variations in that time interval. However, none of those networks was able to outperform the network that achieved the local minimum RMSEf. Therefore, though it is interesting to detect these periodic oscillations, the information regarding the past 20 min observations (4 TDL) has more importance.

The best network performance was achieved with a combination of 5 hidden neurons and a 4 TDL, with the lowest RMSEf and the MAEf being 0.055 and 0.041, respectively. Comparing these results with the results of the best configuration of the NARX neural network using the West

and East PV system information as exogenous inputs (10 hidden neurons and a 2 TDL - RMSEf and MAEf were 0.043 and a 0.031 respectively), one may have noticed a deterioration of about 30% of these final errors.

In fact, the difference of the selected number of hidden neurons for both networks that utilize 2 exogenous inputs might be the result of the many missing observations of the South PV system time series. Moreover, the fact that the North and South PV systems are, geographically, more separated that the West and East PV systems may have influence in the selection of the tapped delay line. Consequently, these variations may explain the fact the network with the West and East PV systems information outperform the network with the North and South PV systems information.

4.5. Case 4 - NARX network with data of 2 meteorological parameters as exogenous inputs.

Selection of the hidden neurons

In this case, very poor accuracy for the multistep predictions was detected while using the default value of 2 TDL to select the best hidden neurons configuration, when comparisons are made to the previous cases. Though in Table 23 the training results appear similar to the previous cases, the multistep predictions process results were rather different than previous cases. The good performance of the training process is due to the fact that a series-parallel configuration is used.

Table 23 - Case 4 training and predicting error results, using different hidden neurons.

Hidden Neurons	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5	0.016	0.089	0.053	0.031	0.124	0.094
10	0.016	0.090	0.054	0.011	0.075	0.057
20	0.016	0.090	0.055	0.026	0.113	0.086
35	0.016	0.090	0.055	0.031	0.121	0.097
50	0.016	0.091	0.055	0.027	0.113	0.097

Table 24 - Case 4 training and predicting coefficients of variation, using different hidden neurons.

Hidden Neurons	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5	1%	0%	2%	29%	14%	10%
10	2%	1%	4%	20%	10%	16%
20	3%	2%	4%	26%	14%	17%
35	3%	2%	5%	45%	23%	28%
50	4%	4%	3%	40%	22%	26%

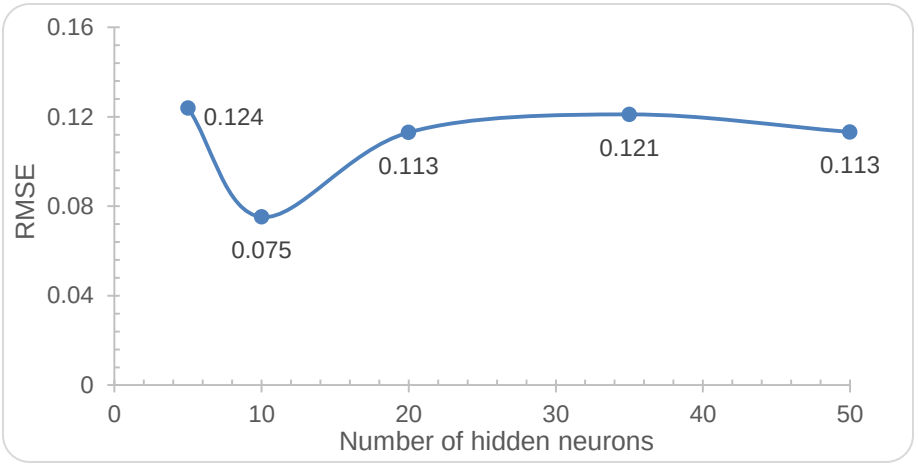

Figure 29 - Case 4 RMSE results for the forecasting process using different hidden neurons.

Interpolating the meteorological data as a preprocessing tool may have had influence in these results, as the network may have not been able to entirely capture the intra hour patterns and effects in the PV system output.

We can observe inTable 23, Table 24 and Figure 29 that the configuration with 10 hidden neurons had the lowest MSEt, RMSEt and MAEt, and therefore was selected to execute further tests. The remainder network configurations either established a too complex architecture or had lack of complexity, and therefore, presented very high multistep forecasting errors when compared to the results of the network with 10 hidden neurons configuration.

Selection of the tapped delay line

The selected the number of hidden neurons revealed significant high RMSEf and MSEf errors while using a 2 TDL in the network configuration. However, Table 25, Table 26 and Figure 30 illustrate that these errors decrease while increasing significantly the tapped delay line value.

Table 25 - Case 4 training and predicting error results, using different TDL.

Real Time	TDL	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
10min	2	0.016	0.090	0.054	0.011	0.075	0.057
20min	4	0.015	0.087	0.053	0.015	0.085	0.062
30min	6	0.015	0.087	0.052	0.019	0.102	0.066
45min	9	0.015	0.087	0.053	0.015	0.090	0.074
1h	12	0.015	0.085	0.052	0.008	0.062	0.047
1h30	16	0.015	0.085	0.052	0.005	0.051	0.038
2h	24	0.014	0.086	0.054	0.005	0.053	0.038
2h30	30	0.014	0.085	0.052	0.005	0.050	0.037
3h	36	0.015	0.085	0.054	0.005	0.051	0.035
4h	48	0.015	0.086	0.055	0.005	0.050	0.036
5h	60	0.015	0.086	0.054	0.006	0.053	0.039

Table 26 - Case 4 training and predicting coefficients of variation, using different TDL.

Real Time	TDL	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
10min	2	2%	1%	4%	20%	10%	16%
20min	4	5%	2%	4%	33%	15%	20%
30min	6	4%	2%	2%	55%	32%	28%
45min	9	2%	1%	2%	38%	22%	31%
1h	12	1%	0%	1%	15%	8%	10%
1h30	16	4%	2%	2%	16%	8%	6%
2h	24	5%	4%	4%	14%	10%	7%
2h30	30	3%	4%	3%	16%	9%	11%
3h	36	4%	2%	5%	19%	11%	8%
4h	48	3%	3%	5%	13%	7%	9%
5h	60	4%	2%	4%	11%	7%	8%

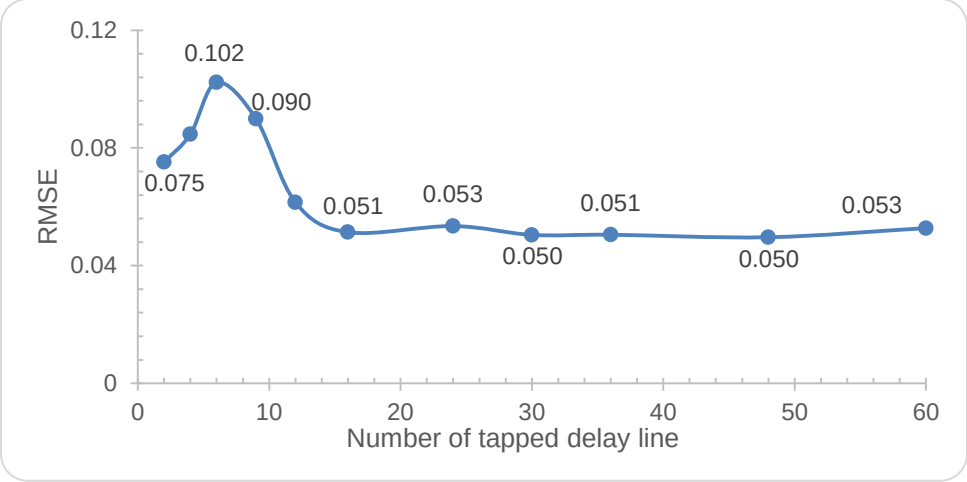

Figure 30 - Case 4 RMSE results for the forecasting process using different TDL.

The error tends to become constant after the 16 TDL (1h30) network configuration; however, minimum values of the RMSEf and MAEf were detected at the 30 TDL (3h) and 36 TDL (4h).Therefore, the best NARX network configuration using meteorological data as exogenous inputs uses a combination of 10 hidden neurons and a 30 TDL.

This 0.050 RMSEf outperforms the previous case where the data from the North and South PV systems used as exogenous inputs yielded a 0.055 RMSEf. However, it was detected that the network performance improved because it kept considering previous outputs, that is, more information regarding the Centre PV system (Output) was fed back to the network. Because the meteorological data did not provide relevant intra-hour information, the parameter that actually influenced the network behavior was the own output.

This case on its own does not pose much interest. However, in the next case all the variables available are used as exogenous inputs and the influence of the combinations of meteorological information and all of the time series of the PV systems is tested.

4.6. Case 5 - NARX network with 4PV systems and meteorological data as exogenous inputs.

Selection of the number of hidden neurons

With Case 5 we tried to understand whether or not adding meteorological information to the information of all PV systems improves the neural network performance. Table 27 clearly shows improvements of the multistep predictions performance comparing to the previous cases, whereas the training process displayed small modifications.

Table 27 - Case 5 training and predicting error results, using different hidden neurons.

Hidden Neurons	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5	0.013	0.081	0.050	0.003	0.040	0.025
10	0.013	0.080	0.050	0.003	0.037	0.025
20	0.013	0.081	0.050	0.004	0.043	0.030
35	0.013	0.082	0.051	0.003	0.042	0.028
50	0.013	0.081	0.053	0.006	0.054	0.043

Table 28 - Case 5 training and predicting error results, using different hidden neurons.

Hidden Neurons	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5	5%	2%	4%	11%	6%	7%
10	5%	2%	4%	15%	10%	11%
20	6%	3%	5%	14%	8%	12%
35	7%	4%	5%	11%	7%	10%
50	7%	4%	6%	21%	11%	22%

The coefficients of variation also kept within the expected range, with the series-parallel architecture showing lower variations than the parallel architecture. Table 27 indicates that the 10 hidden neurons network configuration achieves the lowest RMSEf (0.037) and MAEf (0.025) values. One can observe in Figure 31 a similar behavior as detected in most previous cases. The 5 hidden neurons network configuration has insufficient weights and cannot produce better results than the 10 hidden neurons network configuration. Moreover, the remainder networks performance tend to decay while increasing the number of hidden neurons. Consequently, the 10 hidden neurons were selected to test the networks with different TDL.

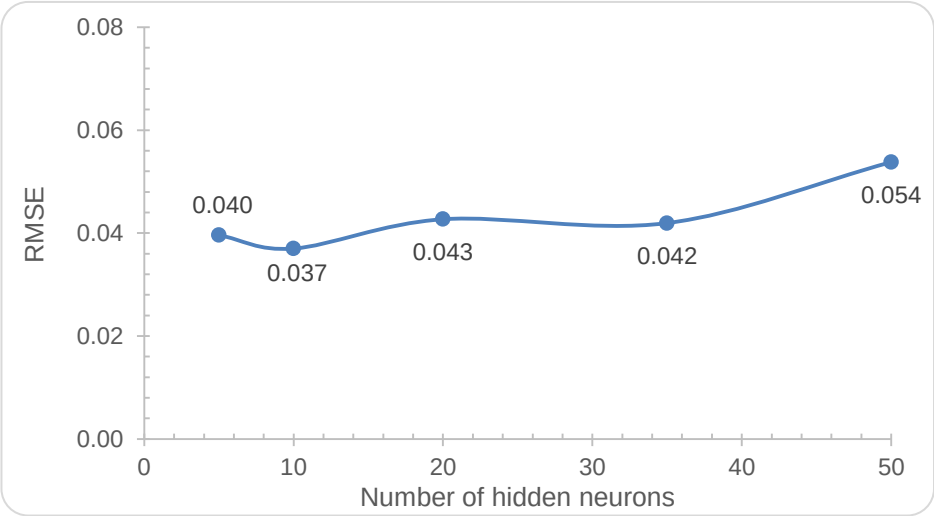

Figure 31 - Case 5 RMSE results for the foecasting process using different hidden neurons.

Selection of the tapped delay line

Varying the tapped delay line did not pose any significant enhancement in the neural network performance. In fact, the error results kept increasing while adding delays to the network. Table 29 clearly indicates that the 10 hidden neurons and 2 TDL neural network performs better than the remainder simulations and is the best suitable architecture to detect the underlying characteristics of the time series studied. Moreover, Table 30 indicates low variability of the 10 hidden neurons and 2 TDL neural network, which indicates very good accuracy.

Table 29 - Case 5 training and predicting error results, using different TDL.

Real Time	TDL	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
10min	2	0.013	0.080	0.050	0.003	0.037	0.025
20min	4	0.013	0.081	0.050	0.004	0.044	0.032
30min	6	0.013	0.083	0.050	0.004	0.044	0.034
45min	9	0.014	0.084	0.052	0.003	0.041	0.030
1h	12	0.014	0.083	0.051	0.004	0.043	0.031
1h30	16	0.014	0.084	0.052	0.004	0.044	0.032
2h	24	0.014	0.085	0.052	0.005	0.048	0.038
3h	36	0.017	0.092	0.059	0.006	0.057	0.042
4h	48	0.018	0.094	0.062	0.007	0.057	0.042

Table 30 - Case 5 training and predicting coefficients of variation, using different TDL.

Real Time	TDL	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
10min	2	5%	2%	4%	11%	6%	7%
20min	4	4%	2%	4%	11%	8%	12%
30min	6	5%	5%	5%	23%	12%	23%
45min	9	7%	4%	6%	14%	8%	11%
1h	12	3%	3%	5%	15%	9%	12%
1h30	16	6%	3%	4%	14%	7%	11%
2h	24	7%	5%	6%	21%	10%	18%
3h	36	11%	6%	8%	32%	18%	19%
4h	48	19%	10%	13%	38%	19%	19%

Furthermore, a thorough observation of Figure 32 led us to understand that is a similar RMSE variation as in previous cases. The global minimum RMSE was, indeed, identified with a 2 TDL network configuration. Although the error of the subsequent tapped delay line is likely to increase, there is usually a sudden drop of the error occurs using around 45min, 1h and up to 2h delayed information. In Figure 32 this drop occurred at the 9 TDL (45 min) network configuration. Thus, though these drops tend to be merely local minimum errors, one can verify that the 45min, 1h and up to 2h delayed information may contain relevant characteristics of the cloud movements. However, most networks revealed better performances with smaller TDL.

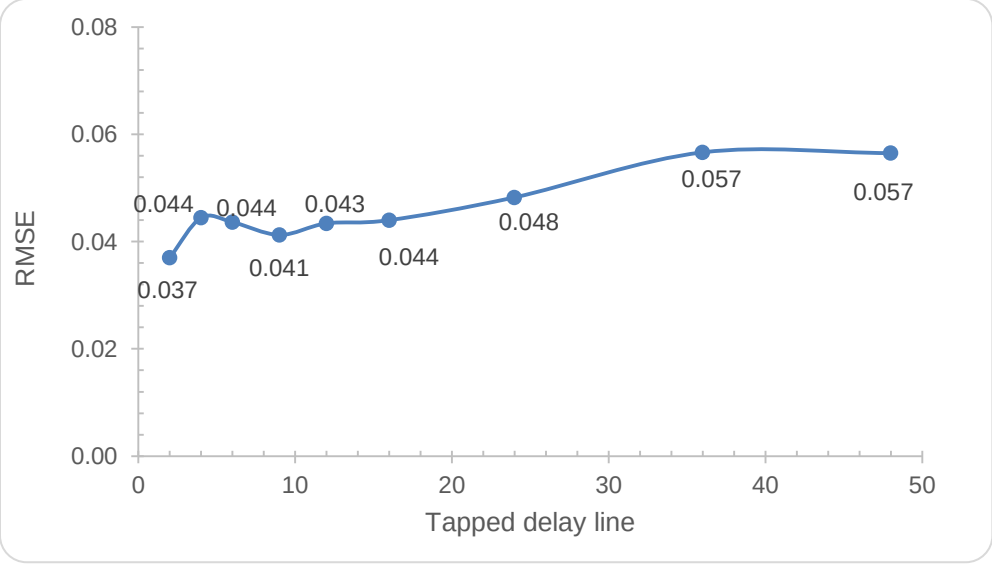

Figure 32 - Case 5, RMSE results for the predicting process using different TDL.

Forecasting results

Considering that a high number of networks configurations were simulated while developing this work, the following figures display the multistep predictions using the best network configuration achieved. This is merely an example of the expected output values using a NARX neural network with 10 hidden neurons, 2 TDL and 6 exogenous inputs.

Figure 33 displays the multistep predictions for the Centre PV system of the last day of the month using the remainder days for test and validation. In Figure 34, one can verify in detail the multistep predictions for that day. The network forecasting results can successfully approximate to the expected outputs and the intra-hour ramping is well captured. However, the network is not able to effectively capture the early and late variations of the day.

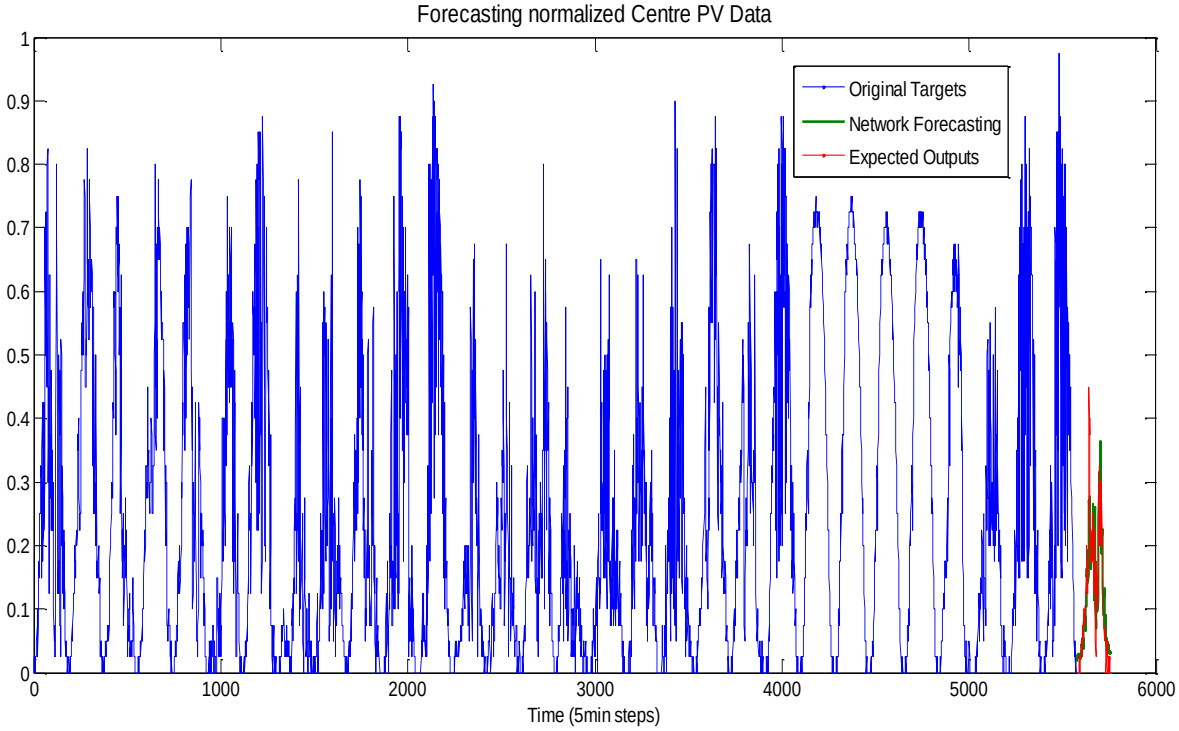

Figure 33 - Prediction of the last day of the month, using NARX network with 4PV systems and meteorological data as exogenous inputs.

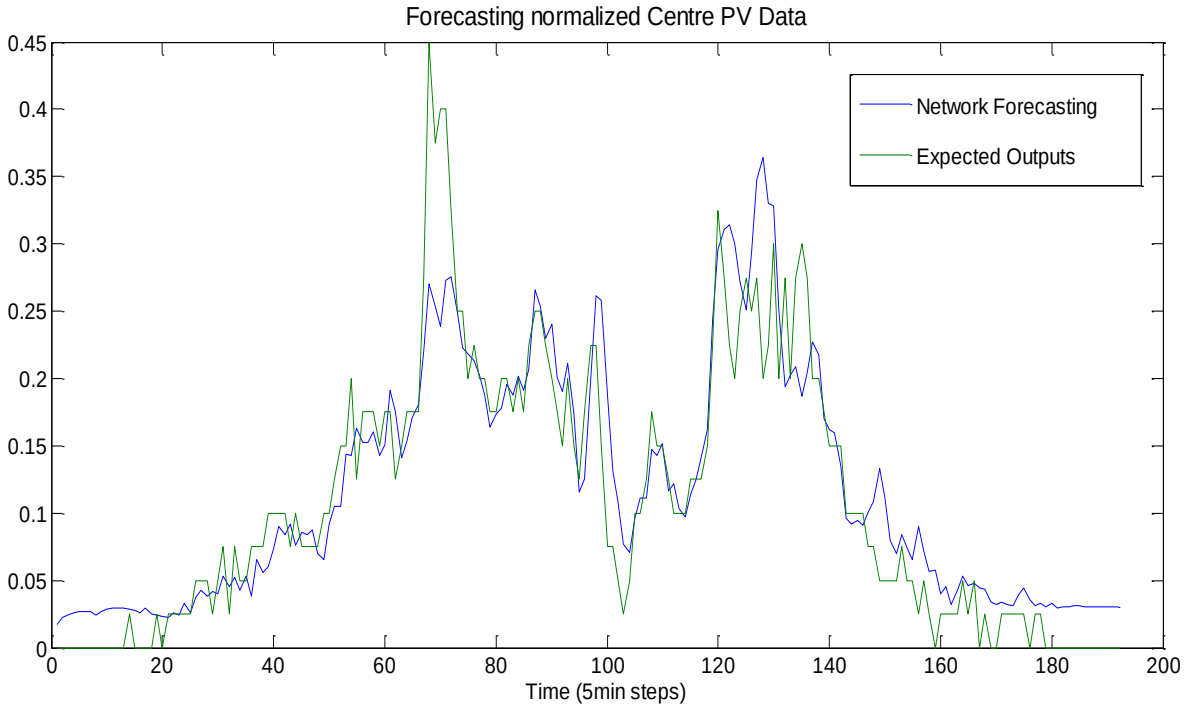

Figure 34 - Comparison between the Network predictions and the expected output for the last day of the month.

Additionally, the following figures illustrate some of the MATLAB$^{@}$ 2012^{Rb} Neural Networks tools to evaluate the training process of only an example of the best network architecture (10 hidden neurons and 2 TDL) achieved in this case.

Figure 35 displays the error autocorrelation function, which is used to validate the network performance. This function describes how the prediction errors are related in time. Ideally, a perfect prediction model, only one nonzero value should occur at the zero lag (this is the mean square error). This would illustrate that the prediction errors were completely uncorrelated with each other (white noise). In this example, the correlations fall approximately within the 95% confidence limits around zero, and therefore, the model seems to be adequate.

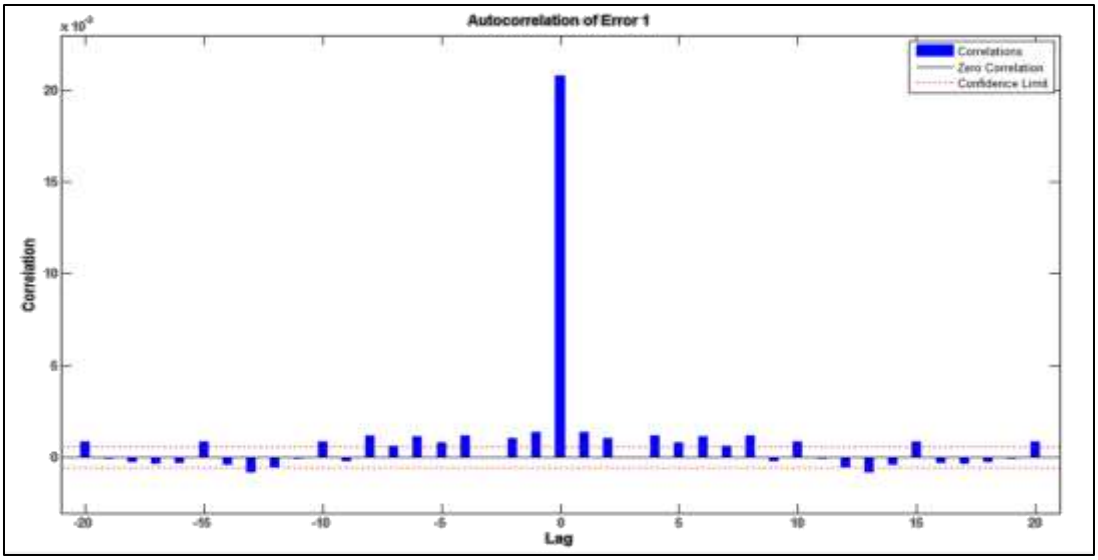

Figure 35 - Error autocorrelation function. The *y* axis has magnitude order of 10^{-3}.

The input-error cross-correlation function, illustrated in Figure 36, indicates how the errors are correlated with input sequence. For a perfect model, the correlations should be zero. However, this model showed good performance due to the fact that all of the correlations fall within the confidence bounds around zero. If this function displayed that the input was correlated with error, the prediction accuracy could have been improved by increasing, for instance, the number of delays in the tapped delay lines.

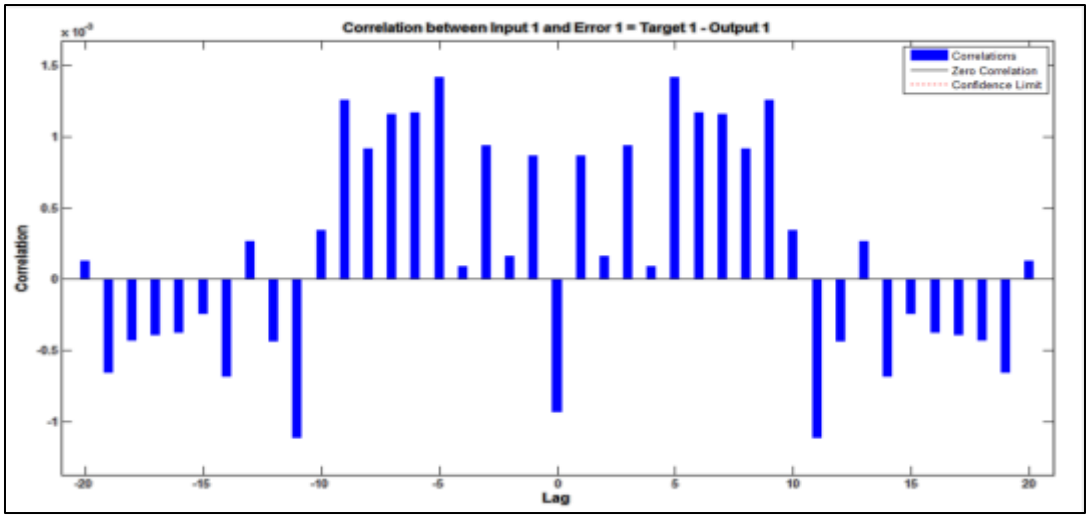

Figure 36 - Input-error cross-correlation. The *y* axis has magnitude order of 10^{-3}.

4.7. Best neural network configurations from Case 2 to 5

Table 31 and Figure 37 illustrate the results of the most suitable network configurations achieved for each case when performing, an equivalent to 1 day, multistep predictions.

Table 31 - Comparison between the optimized networks configurations.

Case	Exogenous Inputs	Output	Hidden neurons	TDL	RMSEf	MAEf
2	4 PV systems (N,S,E,W)	Centre PV system	10	2	0.038	0.027
3.1	2 PV systems – West and East.		10	2	0.043	0.031
3.2	2 PV systems – North and South		5	4	0.055	0.041
4	Meteorological		10	30	0.050	0.037
5	4 PV systems (N,S,E,W) and Meteorological		10	2	0.037	0.025

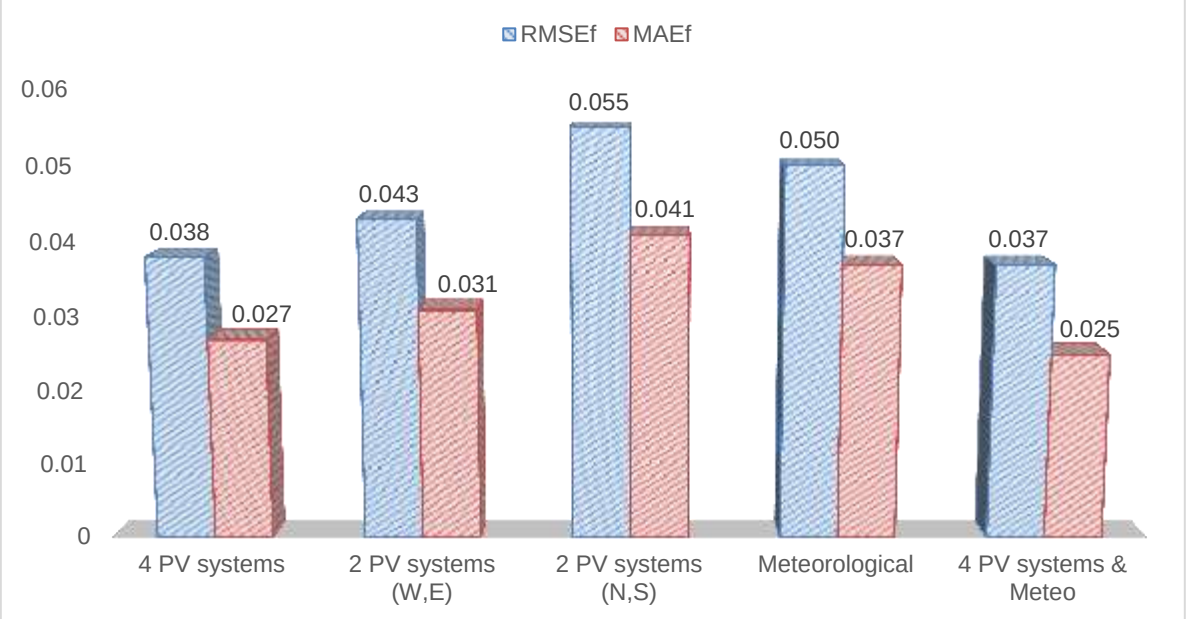

Figure 37 - Comparison between the optimized networks configurations achieved in each case. The *x* label illustrates the input data used in the NARX network.

Case 3.2 presented the highest RMSEf and MAEf results and, as previously mentioned, it was probably a consequence of the large number of missing observations in the South PV system. Furthermore, in case 4 where meteorological data was used as exogenous inputs, a 30 TDL was required to produce lower RMSEf and MAEf values. This indicates the network only performs well when the target series (Centre PV system) past information is strongly fed into the network. However, the neural network in this case was able to achieve a 10% error reduction comparing to case 3.2.

In case 2, case 3.1 and case 5, a 10 hidden neurons and 2 TDL were considered as the best network architecture and produced the three lowest RMSEf and MAEf. Though both NARX neural networks in case 3.1 and case 3.2 use data of 2 PV systems as exogenous inputs, the West and East PV systems combination (case 3.1) clearly outperforms the (North and south) PV systems, with a 30% reduction of the error results. As suggested before, this may be related to missing observations in the South PV system or may be due to a more relevant correlation between the time series of the West and East PV systems with the target system, i.e. predominantly East-West cloud motion. Nevertheless, one can observe that the NARX neural network in case 2 improves the prediction accuracy when combining the 4 PV systems. A 0.038 RMSEf and a 0.027 MAEf pose a 13% and 15%, respectively, accuracy improvement comparing to the case 3.1.

Adding the meteorological data and the 4 PV systems as exogenous inputs (case 5) demonstrated to further improve the NARX neural network performance accuracy with the RMSEf and MAEf decreasing by another 3% and by 8%, respectively. Thus, combining all the information available revealed to be the best method and, therefore, the following experimentation used the case 5 NARX network architecture.

4.8. Case 6 - Multistep ahead forecasting

As mentioned before, in the previous cases the past steps (TDL) of the exogenous inputs and outputs were used to perform several one-step predictions until achieving one complete day of predictions.

In this case, methodologically, and as an example, a 30min prediction ahead (6 steps because each computational step is a 5 min step in real time) using a NARX neural network with 2 TDL consisted in denoting that $t-1$ and $t-2$ values were considered to model the $t+6$ value instead of the $t+1$ value.

Table 32 and Table 33 display the results of the multistep ahead predictions, using a NARX neural network with 10 hidden neurons and 2TDL. Table 34 presents the results of the persistence model and Figure 38 shows the comparison between the NARX model and the persistence model.

Table 32 – Error results of anticipating the predictions using the NARX model.

Time ahead	MSEt	RMSEt	MAEt	MSEf	RMSEf	MAEf
5min	0.013	0.080	0.050	0.003	0.037	0.025
30min	0.027	0.117	0.077	0.008	0.063	0.042
1h	0.027	0.116	0.077	0.007	0.058	0.041
1h30	0.022	0.104	0.069	0.005	0.052	0.036
2h	0.020	0.101	0.065	0.004	0.046	0.032
2h30	0.021	0.104	0.069	0.004	0.045	0.034
3h	0.026	0.114	0.079	0.007	0.059	0.044
4h	0.046	0.152	0.113	0.022	0.104	0.081
6h	0.074	0.192	0.146	0.030	0.124	0.101
8h	0.081	0.208	0.157	0.040	0.144	0.116
12h	0.072	0.190	0.145	0.034	0.135	0.109
16h	0.059	0.173	0.126	0.060	0.175	0.134

Table 33 - CV results of anticipating the predictions with different intervals.

Time ahead	CV_{MSEt}	CV_{RMSEt}	CV_{MAEt}	CV_{MSEf}	CV_{RMSEf}	CV_{MAEf}
5min	5%	2%	4%	11%	6%	7%
30min	4%	3%	3%	8%	11%	7%
1h	4%	3%	3%	12%	6%	8%
1h30	6%	3%	5%	16%	9%	8%
2h	5%	4%	6%	12%	7%	8%
2h30	5%	3%	5%	4%	3%	4%
3h	11%	6%	8%	14%	7%	8%
4h	7%	4%	3%	16%	9%	8%
6h	7%	4%	3%	16%	9%	8%
8h	4%	6%	8%	26%	20%	15%
12h	10%	5%	6%	11%	12%	9%
16h	8%	5%	6%	15%	6%	8%

Table 34 - Error results of anticipating the predictions using the persistence model.

Time ahead	MSEf	RMSEf	MAEf
5min	0.00	0.05	0.03
30min	0.01	0.07	0.05
1h	0.01	0.09	0.06
1h30	0.01	0.11	0.08
2h	0.02	0.13	0.10
2h30	0.02	0.14	0.11
3h	0.03	0.16	0.13
4h	0.04	0.20	0.15
6h	0.07	0.27	0.21
8h	0.11	0.32	0.24
12h	0.09	0.31	0.22
16h	0.09	0.31	0.21

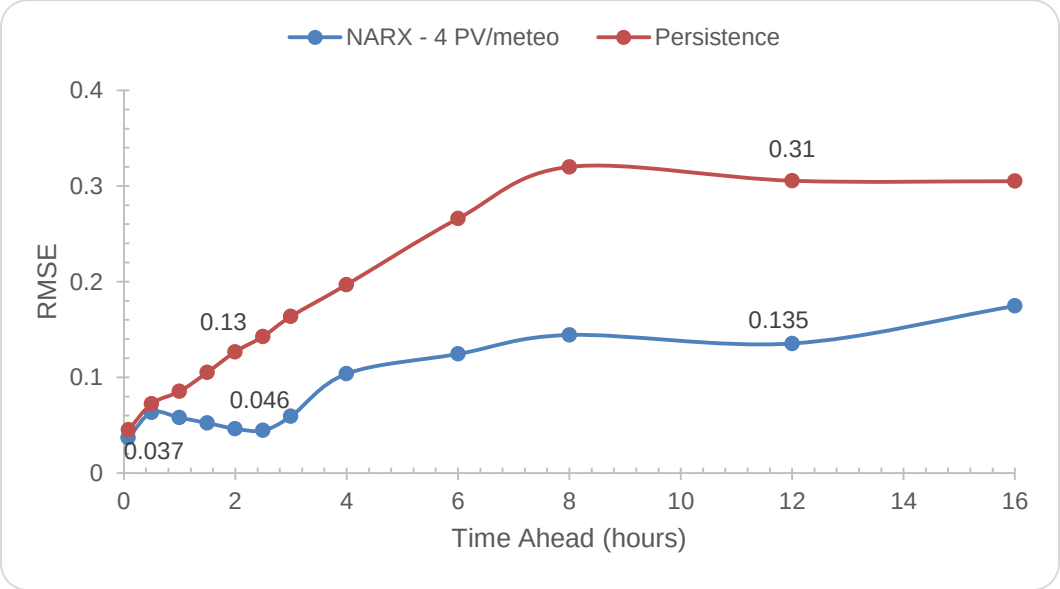

Figure 38 - Comparison between the persistence model and the NARX model RMSEf results of anticipating the predictions.

In Figure 37, one can detect that the global minimum errors of both models occur with a single step ahead predictions (5min), albeit the NARX model (0.037 RMSE) outperforms the persistence model (0.05 RMSE). In fact, the persistence model cannot outperform the NARX model

at any time because as expected, its RMSE value severely increases while increasing the time step ahead predictions.

Furthermore, the NARX results show that anticipating the predictions between 2h and 2h30 generates a very interesting performance, with the local minimum of 0.046 (RMSEf) and 0.032 (MAEf) being reached when performing a 2h30 forecasting. This led us to believe that a current value may have significant influence in a 2h30 later observation. In fact, though forecasting errors between 30 min and 2h were not extremely high, they did not outperform the 2h30 forecasting results.

One can observe in Figure 38 that the forecast accuracy dramatically decays when the forecast horizon is increased beyond the 2h horizon. Indeed, the difference between the accuracy of the NARX neural network performance for 16 hours forecast horizon and the 2h30 forecasting horizon is very significant.

Finally, it is relevant to highlight the fact that the tested scenarios in case 6 used a 10 hidden neurons and 2 TDL configuration for the NARX model. This configuration was selected based on the one-step forecasts performed in case 5, which indicates that the network was optimized for the case 5 conditions. Thus, in Figure 38, the results beyond the 5min forecasts are not representative of optimized NARX networks. Naturally, the training process (Table 32) illustrates this statement, as there are some variations between the different trials. However, the NARX model depicts an acceptable behaviour and a significant enhancement relatively to the persistence model. Optimizing the networks would probably result in improving the network performances and consequently the RMSE values in Figure 38.

5. CONCLUSION

This work provides a thorough research in the solar power forecasting domain focusing on time series prediction using the NARX model. Accordingly, a NARX model using a Feedforward network for the training process and a recurrent network for forecasting is introduced to optimize the final performance.

Several combinations of data from 4 PV systems geographically separated from each other, and meteorological data from the area of Utrecht, Netherlands, are proposed as exogenous inputs. This allowed us to understand what parameters are more relevant to the forecast performance of a PV system located in the middle of other 4 PV systems, while using the same NARX model. Every combination requires testing different architectures and, thus, selecting the optimized network. To analyse the performance of the several cases different evaluator functions are introduced. By observing the MSE, RMSE and MAE values it was possible to select the most appropriate architecture and parameters.

It was proved that the information of 4 PV systems as the network inputs outperforms networks solely using 2 time series of PV systems as inputs. Therefore, it was shown that considering more information of neighbouring distributed PV systems can enhance the forecasts accuracy. Moreover, adding the meteorological information to the network using the information of the 4 PV systems has proved to further improve the accuracy.

Furthermore, using the best network configuration, several forecasting horizons were tested to analyse the NARX network ability to forecast in advance. The NARX model clearly outperforms the persistence model even for very small horizons, which indicates that considering recent past observations is a better method to perform predictions.

For the NARX model, anticipating the 1 day prediction by 5min (3.7% RMSE) and 2h30 (4.5% RMSE) proved to be the global and local minimums, respectively, for the input and output data considered. Thus, we can also determine that for the PV system output and for the day predicted, a current value of the PV system shows very good ability to recognize patterns of the 2h30 previous observation. The NARX model is highly effective in multistep predicting but showed that these predictions were more accurate by anticipating them up to 2h30.

Though the model was able to accurately capture the intra-hour solar ramps, it also had some limitations, such as the length of the time series. Indeed, neural networks may require historical data that sometimes is not available.

To overcome the single limitations of a model, hybrid models that use different models features to capture different patterns in the data may improve the forecast performance. Therefore, neural networks associated with other models, such as the total sky imagery or satellite based models, could be a fundamental instrument to improve intra-day forecasts accuracy. The intra-day horizon is currently of smaller economic value than the day-ahead forecasts; however, substantial market opportunities will likely materialize by increasing the solar penetration and improving the accuracy of intra-day forecasts.

It is highly fundamental to foster the joint efforts between system operators and the research community, so that novel approaches adapted to the operation of grid systems with a strong presence of variable renewables can be implemented. Though forecasting fast solar ramps has not yet raised enough attention from the research community, the introduction of the smart grid with predictive control of buildings and electricity loads will place its own requirements on solar and PV forecasting and help incentive new developments.

6. REFERENCES

Bacher, P.; Madsen, H.; Nielsen, H.A.; & Plads, R. (2009). Online short-term solar power forecasting. Solar Energy,83(10), pp. 1772–1783.

Beale, M.; Hagan, M.; & Demuth, H. (2013). Neural Networks Toolbox - User's guide. Massachusetts: Mathworks.

Bie, T.; & Musikowski, H. (2008). Forecasting photovoltaic energy using a fourier series based method. Valencia (Spain): 23rd European Photovoltaic Solar Energy Conference, pages 3088–3091.

Box, G., Jenkins, G., & Reinsel, G. (1994). Time Series Analysis - Forecasting and control (Third Edition ed.). New Jersey: Prentice-Hall International, Inc.

Boxwell, M. (2013). Solar Electricity Handbook. Warwickshire: Greenstream Publishing.

Bramer, M. (2006). Artificial Intelligence in Theory and Practice. Santiago, Chile: Springer.

Brinkworth, B. (1977). Autocorrelation and stochastic modelling of insolation sequences. 19(4), pp. 343–347.

Chaouachi, A.; Kamel, R.M.; Ichikawa, R.; Hayashi, H.; & Nagasaka, K. (2009). Neural Network Ensemble-based Solar Power Generation Short-Term Forecasting. International Journal of Information and Mathematical Sciences, 30.

Coit, D; Jackson, B.T; & Smith, A.E. (1997). Static Neural Network Process Models: Considerations and case studies. International Journal of Production Research.

Cornaro, C., Bucci, F., Pierro, M., Del Frate, F., Peronaci, S., & Taravat, A. (2009). Solar Radiation forecast using Neural Networks for the Prediction of Grid. World Academy of Science, Engineering and Technology.

Dazhi, Y.; Jirutitijaroen, P.; & Walsh, W.M. (2012). Hourly solar irradiance time series forecasting using cloud cover index. Solar Energy, 86(3531–3543).

Di Piazza, A., Di Piazza, M., & Vitale, G. (2013). Solar Radiation Estimate and Forecasting by Neural Networks-based Approach.

Diaconescu, E. (2008). The use of NARX Neural Networks to predict Chaotic Time Series. Volume 3, pp. Pages 182-191.

Diagne, H.; David, M.; & Boland, J. (2012). Solar irradiation forecasting: State-of-the-art and proposition for future developments for small-scale insular grids. American Solar Energy Society.

Dreyfus, G. (2005). Neural Networks Methodology and Applications. Paris, France: Springer.

Espinar, B.; Aznarte, J.; Girard, R.; Moussa, A. & Kariniotakis, G. (2010). Photovoltaic Forecasting: A state of the art. Tarragona, Spain: 5th European PV-Hybrid and Mini-Grid Conference.

Hammer, A; Heinemann, D; Hoyer-Klick, C; Lorenz, E; Mayer, B; & Schroedter-Homscheidt, M. (2007). Remote Sensing and Atmospheric Physics for an Efficient Use of Renewable Energies. Virtual Institute for Energy Meteorology.

Haylin, S. (1999). Neural Networks - A Comprehensive foundation. Hamilton, Ontario, Canada: Prentice Hall International, Inc.

Heimo, A.; Sempreviva, A.M.; Kuik, F.; & Gryning, E. (August 2012). Weather Intelligence for Renewable Energies (WIRE) – Current state Report.

Hill, T., Marquez, L., O'Connor, M., & Remus, W. (1994). Artificial neural networks for forecasting and decision making. International Journal of Forecasting.

Hoff, T., & Perez, P. (2012). Predicting short-term variability of high-penetration PV. American Solar Energy Society – Proc. ASES Annual Conference, Raleigh, NC.

Inman, R., Pedro, T., & Coimbra, C. (2013). Solar forecasting methods for renewable energy integration. Progress in Energy and Combustion Science, 39.

Jain, A.K.; Mao, J. (1996). Artificial Neural Networks: A tutorial. Michigan State.

Jayadevan, V.T.; Rodriguez, J.J.; Lonij, V.P.A.; & Cronin, A.D. (2012). Forecasting solar power intermittency using ground-based cloud imaging. ASES World Renewable Energy Forum proceedings.

Krenker, A.; Bešte, J.; & Kos, A. (2011). Introduction to the Artificial Neural Networks. Ljubljana.

Kriesel, D. (2005). A brief introduction to Neural Networks.

Kudo, M; Takeuchi, A; Nozaki, Y; End, H; & Sumita, J. (2009). Forecasting Electric Power Generation in a Photovoltaic Power Systems for an Energy Network. Electrical Engineering in Japan, Vol. 167, No. 4, pages 16–23.

Lorenz, E.; Hurka, J.; Karampela, D.; Beyer, H.; & Schneider, M. (2008). Qualified forecast of ensemble power production by spatially dispersed grid-connected PV systems.

Mantzari, V., & Mantzaris, D. (2013). Solar radiation: Cloudiness forecasting using a soft computing approach. Artificial Intelligence Research, Vol. 2, No. 1.

Marquez, R., & Coimbra, C. (2011). Forecasting of global and direct solar irradiance using stochastic learning methods, ground experiments and the NWS database. Solar Energy, 85 746-756.

Marquez, R.; Coimbra, F.M.C. (2012). Intra-hour DNI forecasting based on cloud tracking image analysis.

Menezes, J., & Barreto, G. (2008). Long-term time series prediction with the NARX network: An empirical evaluation. Neurocomputing 71, 3335–3343.

Mihalakakou, G., Santamouris, M., & Asimakopoulos, D. (2000). The total solar radiation time series simulation in Athens, using neural networks. Theoretical and Applied Climatology, 66 pp. 185–197.

Nelson, M., Hill, T., Remus, B., & O'Connor, M. (1994). Can neural networks be applied to time series forecasting and learn seasonal patterns: an empirical investigation.

Palit, A., & Popovic, D. (2005). Computacional Intelligence in Time Series Forecasting: Theory and Engineering Applications. Springer.

Paoli, C., Voyant, C., Muselli, M., & Nivet, M. (2010). Forecasting of preprocessed daily solar radiation time series, using neural networks. Solar Energy, 2146–2160.

Pelland, S.; Remund, J.; Kleissl, J; Oozeki, T.; & De Brabandere, K. (2013). Photovoltaic and Solar forecasting - State of the art.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Universität Karlsruhe, Karlsruhe, Germany: Neural Network.

Puri, V. (1978). Estimation of half-hour solar radiation values from hourly values. Solar Energy, Volume 21(Issue 5), pp. Pages 409–414.

Remund, J.; Perez, R. ; & Lorenz, E. (2008). Comparison of solar radiation forecasts for the USA. 23rd European Photovoltaic Solar Energy Conference.

Rikos, E.; Tselepis, S.; Hoyer-Klick, C.; & Schroedter-Homscheidt, M. (2008). Stability and Power Quality Issues in Microgrids Under Weather Disturbances.

Sfetsos, A., & Coonick, A. (2000). Univariate and Multivariate forecasting of hourly solar radiation with artificial intelligence techniques. Solar Energy, 68 pp. 169–178.

Sharda, R., & Patil, R. (1992). Connectionist approach to time series prediction on Neural Networks. Journal of Intelligent Manufacturing.

Stapleton, G., & Neill, S. (2012). Grid-Connected Solar Electric Systems. New York: Earthscan.

Tan, C. (2009). Financial Time Series Forecasting Using Improved Wavelet Neural Network.

Tang, Z., & Fishwick, P. (1993). Feedfowrard neural networks as models for time series forecasting. Journal on computing, 374-385.

Twidell, J., & Weir, T. (2006). Renewable Energy Resources. New York: Taylor & Francis.

Weigend, A., Hubernam, B., & Rumelhart, D. (1992). Predicting sunspots and exhance rates with connectionist network. 385-432(Nonlinear modeling and forecasting).

Yona, A., Senjyu, T., Saber, A., & Funabashi, T. (2007). Application of Neural Network to One-Day-Ahead 24 hours Generating Power Forecasting for Photovoltaic System.

Yu, H., & Wilamowski, B. (2010). Levenberg-Marquardt Training. Auburn University.

Zeng, J. & Qiao, Wei. (2011). Short-Term Solar Power Prediction Using an RBF Neural Network. San Diego: Power and Energy Society General Meeting.

Zhang, G., Eddy Patuwo, B., & Hu, M. (1997). Forecasting with artificial neural networks. International Journal of Forecasting.

Zweibel, K. (1982). Basic Photovoltaic Principles and Methods. Colorado: Technical Information Office.

7. ANNEX - Matlab Code

```
%% 1. Importing data and preprocessing
clc; clear all; close all;
x=load('paineis_1min.txt');

x1=x(360:length(x),:);%remove 1st 360zeros of the 1st of july

x_semzeros=zeros(28471,5);
 dia = 930; noite = 510; diasmes = 31;%define day as 930steps and night as 510steps (1min time series)
 for col = 1:5
    for lin = 1:diasmes
        x_semzeros(dia*(lin-1)+1:dia*lin,col) = x1((lin-1)*(dia+noite)+1:(lin-1)*(dia+noite)+dia,col);%
removes all “night values”
    end
end


%MOVING AVERAGE FUNCTION
function [y]=mediaMovel(Sinal,Nfiltro)
h=1/Nfiltro;
n=length(Sinal);
m=(Nfiltro-1)/2;
aux=zeros(n+(2*m),1);%creates zero matrix
for i=1+m:n+m %defines auxiliar vector (aux)
    aux(i)=Sinal(i-m);
end

for i=1:n
    suma=0;
    for k=-m:m
        suma=suma+aux((i+m)-k)*h;
    end
    y(i)=suma;
end
end

Centro=x_semzeros(:,1);West=x_semzeros(:,2);North=x_semzeros(:,3);East=x_semzeros(:,4);South=x_semz
eros(:,5);%defines every column.
%figure
% subplot(132)
% plot(Centro,'b');hold on;plot(West,'r');plot(North,'m');plot(East,'c'); plot(South,'y');
% title('PV - day cycles'),xlabel('time (min)'),ylabel('Power
Watts)'),legend('Centro','West','North','East','South');


%Normalize each series from 0 to 1
CentroN = (Centro - min(Centro)) / ( max(Centro) - min(Centro) );
WestN =(West - min(West)) / ( max(West) - min(West) );
NorthN = (North - min(North)) / ( max(North) - min(North) );
EastN = (East - min(East)) / ( max(East) - min(East) );
```

```
SouthN = (South - min(South)) / ( max(South) - min(South) );
%figure
% subplot(133)
% plot(CentroN,'b');hold on;plot(WestN,'r');plot(NorthN,'m');plot(EastN,'c'); plot(SouthN,'y');
% title('Normalized PV data - day cycles'),xlabel('Time
min)'),ylabel('');legend('Centro','West','North','East','South');


% % %Scales the series to [-1,1]
 ymax = 1;ymin = -1;
CentroN = (ymax-ymin)*(Centro-min(Centro))/(max(Centro)-min(Centro))+ymin;
WestN = (ymax-ymin)*(West-min(West))/(max(West)-min(West))+ymin;
NorthN = (ymax-ymin)*(North-min(North))/(max(North)-min(North))+ymin;
EastN = (ymax-ymin)*(East-min(East))/(max(East)-min(East))+ymin;
SouthN = (ymax-ymin)*(South-min(South))/(max(South)-min(South))+ymin;


%Load meteorological data
meteoaux=load('dadosmeteo.txt');
Temp=meteoaux(:,1);Rad=meteoaux(:,2);
% figure
% subplot(211);plot(Temp),hold on,title('Temperature Raw data'),xlabel('Time (hours)'),ylabel('ºC')
% subplot(212);plot(Rad),hold on,title('Radiation Raw data'),xlabel('Time (hours)'),ylabel('J/cm2')

%INTERPOLATION MIN to MIN
TempInterp = interp(Temp,60);RadInterp = interp(Rad,60);
TempInterp(TempInterp<0)=0;RadInterp(RadInterp<0)=0;
 meteodata = [TempInterp,RadInterp];

x2 = meteodata(360:length(x),:);
x_semzeros2 = zeros(28471,2);
dia = 930; noite = 510; diasmes = 31;

for col = 1:2
   for lin = 1:diasmes
      x_semzeros2(dia*(lin-1)+1:dia*lin,col) = x2((lin-1)*(dia+noite)+1:(lin-1)*(dia+noite)+dia,col);
   end
end

Temperature = x_semzeros2(:,1);
Radiation = x_semzeros2(:,2);

% Scale Meteorological data [-1,1]
TempN =(ymax-ymin)*(Temperature-min(Temperature))/(max(Temperature)-min(Temperature))+ymin;
RadN = (ymax-ymin)*(Radiation-min(Radiation))/(max(Radiation)-min(Radiation))+ymin;

TempN=(Temperature - min(Temperature)) / ( max(Temperature) - min(Temperature) );
RadN =(Radiation - min(Radiation)) / ( max(Radiation) - min(Radiation) );

%Normalized vector ( -1 a 1)
xN=[CentroN,WestN,NorthN,EastN,SouthN,TempN,RadN];
```

```
 % Applying the Moving Average(MA) – 5min,10min,30min,1h

%Converting to 5min step time series with 5min MA
P1=mediaMovel(CentroN,5);P2=mediaMovel(WestN,5);
P3=mediaMovel(NorthN,5);P4=mediaMovel(EastN,5);
P5=mediaMovel(SouthN,5);P6=mediaMovel(TempN,5);
P7=mediaMovel(RadN,5);

P1_=P1(1:5:end);P2_=P2(1:5:end);
P3_=P3(1:5:end);P4_=P4(1:5:end);
P5_=P5(1:5:end);P6_=P6(1:5:end);P7_=P7(1:5:end);%Remove data with a 5step interval
p5em5=[P1_', P2_', P3_',P4_', P5_',P6_',P7_'];)

% %Converting to 15min step time series with 15min MA
P1a=mediaMovel(CentroN,15);P2a=mediaMovel(WestN,15);
P3a=mediaMovel(NorthN,15);P4a=mediaMovel(EastN,15);
P5a=mediaMovel(SouthN,15);P6a=mediaMovel(TempN,15);
P7a=mediaMovel(RadN,15);%faz media movel de 5min

P1a_=P1a(1:15:end);P2a_=P2a(1:15:end);
P3a_=P3a(1:15:end);P4a_=P4a(1:15:end);
P5a_=P5a(1:15:end);P6a_=P6a(1:15:end);
P7a_=P7a(1:15:end);%Retirar dados de 15em15
 p15em15=[P1a_', P2a_', P3a_', P4a_', P5a_',P6a_',P7a_'];

% %%Converting to 30min step time series with 30min MA
% P1b=mediaMovel(CentroN,31);P2b=mediaMovel(WestN,31);
P3b=mediaMovel(NorthN,31);P4b=mediaMovel(EastN,31);
P5b=mediaMovel(SouthN,31);P6b=mediaMovel(TempN,31);
P7b=mediaMovel(RadN,31);%faz media movel de 5min

P1b_=P1b(1:30:end);P2b_=P2b(1:30:end);
P3b_=P3b(1:30:end);P4b_=P4b(1:30:end);
P5b_=P5b(1:30:end);P6b_=P6b(1:30:end);
P7b_=P7b(1:30:end);%Retirar dados de 30em30 minutos
% p30em30=[P1b_', P2b_', P3b_', P4b_', P5b_',P6b_',P7b_'];

% %%Converting to 1h step time series with 1h MA

P1c=mediaMovel(CentroN,61);P2c=mediaMovel(WestN,61);
P3c=mediaMovel(NorthN,61);P4c=mediaMovel(EastN,61);
P5c=mediaMovel(SouthN,61);P6c=mediaMovel(TempN,61);
P7c=mediaMovel(RadN,61);

P1c_=P1c(1:60:end);P2c_=P2c(1:60:end);
P3c_=P3c(1:60:end);P4c_=P4c(1:60:end);
P5c_=P5c(1:60:end);P6c_=P6c(1:60:end);
P7c_=P7c(1:60:end);%Retirar dados de 30em30 minutos
% p1hem1h=[P1c_', P2c_', P3c_', P4c_', P5c_',P6c_',P7c_'];


%% 2. Data preparation
```

```
N1=5766; %length of the time series with 5 min step
N = 192; % Multi-step prediction (1 day) – 5min step

% Input and target series are divided in two groups of data:
% 1st group: used to train the network
inputseries  = p5em5(1:N1-N,2:7);
targetseries = p5em5(1:N1-N,1);

% 2nd group: this is the new data used for simulation.
%inputSeriesVal will be used for predicting new targets. targetSeriesVal will be used for network validation
after prediction
inputseriesVal  = p5em5(N1-N+1:N1,2:7);
targetseriesVal = p5em5(N1-N+1:N1,1); % This is generally not available

inputSeries = tonndata(inputseries,false,false);
targetSeries = tonndata(targetseries,false,false);
inputSeriesVal = tonndata(inputseriesVal,false,false);
targetSeriesVal = tonndata(targetseriesVal,false,false);
```

%% 3. Network Architecture

```
% Create a Nonlinear Autoregressive Network with External Input
delay = 2; %number of tapped delays

jj=0;
for neuronsHiddenLayer = 10 %Number of Neurons in the Hidden Layer
jj=jj+1;

% Network Creation
Ntrial = 1;%number of training trials

for ji = 1: Ntrial %Number of tests

   net = narxnet(1:delay,1:delay,neuronsHiddenLayer);
```

%% 4. Training the network

```
 [Xsinputs,XiinputStates,AilayerStates,Tstargets] = preparets(net,inputSeries,{},targetSeries);

% Customize training parameters
net.trainFcn = 'trainlm';  % Levenberg-Marquardtalgotihm
net.trainParam.epochs = 1000;
net.divideFcn = 'divideblock';
net.divideParam.trainRatio = 60/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 20/100;

% Choose a Performance Function:
net.performFcn = 'mse';  % Mean squared error

%activation functions
```

```
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';

% Train the Network
[net,tr] = train(net,Xsinputs,Tstargets,XiinputStates,AilayerStates);%training the network

Y = net(Xsinputs,XiinputStates,AilayerStates);%Simluations

% Performance for the series-parallel implementation, only one-step-ahead prediction
errors = gsubtract(Tstargets,Y);

%Results analysis - Series-parallel
MSEt(ji,jj) = mse(net,Tstargets,Y);%mean square error
RMSEt(ji,jj) = sqrt(MSEt(ji,jj));%root mean square error
MAEt(ji,jj) = mae(net,Tstargets,Y);%mean absolute error

% 5. Multi-step ahead prediction

inputSeriesPred  = [inputSeries(end-delay+1:end),inputSeriesVal];
targetSeriesPred = [targetSeries(end-delay+1:end),con2seq(nan(1,N))];
netc = closeloop(net);%starts the feedback process
%view(netc)

[Xsinputs,XiinputStates,AilayerStates,Tstargets] = preparets(netc,inputSeriesPred,{},targetSeriesPred);
yPred = netc(Xsinputs,XiinputStates,AilayerStates);

% FORECASTING Results analysis
MSEf(ji,jj) = mse(netc,targetSeriesVal,yPred)
RMSEf(ji,jj) = sqrt(MSEf(ji,jj))
MAEf(ji,jj) = mae(netc,targetSeriesVal,yPred)

end
end

%TRAINING ASSESSMENT PART
%Delete outliers

MSEtt = sort(MSEt);%displays the results from the lowest to the highest
mediana_MSEt = median(MSEtt);%calculates the median of the sample
MSEt(MSEt>1.15*mediana_MSEt)=[];%removes all values 15%higher than the median value

RMSEtt = sort(RMSEt);
mediana_RMSEt = median(RMSEtt);
RMSEt(RMSEt>1.15*mediana_RMSEt)=[];

MAEtt = sort(MAEt);
mediana_MAEt = median(MAEtt);
MAEt(MAEt>1.15*mediana_MAEt)=[];

media_MSEt = mean((MSEt));%mean of the Ntrials for each Neuron
sigma_MSEt = std(MSEt);%standard deviation
CV_MSEt = sigma_MSEt/media_MSEt;
```

```
media_RMSEt = mean(RMSEt);%mean of the Ntrials for each Neuron
sigma_RMSEt = std(RMSEt);%standard deviation
CV_RMSEt = sigma_RMSEt/media_RMSEt;

media_MAEt = mean((MAEt));%mean of the Ntrials for each Neuron
sigma_MAEt = std(MAEt);%standard deviation
CV_MAEt = sigma_MAEt/media_MAEt;

% %FORECASTING ASSESSMENT PART
% %Delete outliers
MSE = sort(MSEf);
mediana_MSE = median(MSE);
MSEf(MSEf>1.15*mediana_MSE)=[];

RMSE = sort(RMSEf);
mediana_RMSE = median(RMSE);
RMSEf(RMSEf>1.15*mediana_RMSE)=[];

MAE = sort(MAEf);
mediana_MAE = median(MAE);
MAEf(MAEf>1.15*mediana_MAE)=[];

media_MSEf = mean((MSEf));%mean of the Ntrials
sigma_MSEf = std(MSEf);%standard deviation
CV_MSEf = sigma_MSEf/media_MSEf;

media_RMSEf = mean(RMSEf);%mean of the Ntrials
sigma_RMSEf = std(RMSEf);%standard deviation
CV_RMSEf = sigma_RMSEf/media_RMSEf;

media_MAEf = mean((MAEf));%mean of the Ntrials
sigma_MAEf = std(MAEf);%standard deviation
CV_MAEf = sigma_MAEf/media_MAEf;

%Final Table
Erro_mediat = [media_MSEt',media_RMSEt',media_MAEt']
% sigma_mediat = [sigma_MSEt',sigma_RMSEt',sigma_MAEt',,sigma_MAPEt']
CV_mediat = [CV_MSEt',CV_RMSEt',CV_MAEt']

%FORECASTING
Erro_mediaf = [media_MSEf',media_RMSEf',media_MAEf']
%sigma_mediaf = [sigma_MSEf',sigma_MAEf',sigma_RMSEf',sigma_MAPEf']
CV_mediaf = [CV_MSEf',CV_RMSEf',CV_MAEf']


%CREATE TABLE
f1 = figure ('name','Table of Errors', 'Position', [100 100 600 200]);
dados = {Erro_mediat(1,1),Erro_mediat(1,2),Erro_mediat(1,3),...
    Erro_mediaf(1,1),Erro_mediaf(1,2),Erro_mediaf(1,3),...
    CV_mediat(1,1),CV_mediat(1,2),CV_mediat(1,3),...
    CV_mediaf(1,1),CV_mediaf(1,2),CV_mediaf(1,3)};
```

```
cname = {'MSEt', 'RMSEt', 'MAEt', ....
   'MSEf', 'RMSEf', 'MAEf', ....
   'CV_MSEt','CV_MAEt','CV_RMSEt',...
   'CV_MSEf','CV_MAEf','CV_RMSEf'};

rname = {'5','20','35','50','65'};
t1 = uitable('Parent',f1,'Data',dados, 'ColumnName', cname,'RowName',rname, 'position', [20 20 360
100]);

%CONVERT [-1,1] to [0,1]
% targetSeries_rev = ((cell2mat(targetSeries)) - ymin) / (ymax-ymin)* (max(cell2mat(targetSeries))-
min(cell2mat(targetSeries))) + min(cell2mat(targetSeries));

% targetSeriesVal_rev =((cell2mat(targetSeriesVal)) - ymin) / (ymax-
ymin)*(max(cell2mat(targetSeriesVal)) - min(cell2mat(targetSeriesVal))) +
min(cell2mat(targetSeriesVal)));

% yPred_rev = ((cell2mat(yPred) - ymin) /(ymax-ymin))*(max(cell2mat(yPred)) -
min(cell2mat(yPred)) + min(cell2mat(yPred)));

figure
plot([cell2mat(targetSeries),nan(1,N);
     nan(1,length(targetSeries)),cell2mat(yPred);
     nan(1,length(targetSeries)),cell2mat(targetSeriesVal)]')

title('Normalized Centre PV Data'),xlabel('Time (5min steps'),ylabel('')
legend('Original Targets','Network Forecasting','Expected Outputs')

figure
plot([cell2mat(yPred);cell2mat(targetSeriesVal)]')

title('Normalized Centre PV Data'),xlabel('Time (5min steps)'),ylabel('')
legend('Network Forecasting','Expected Outputs')
```