2016 IEEE International Conference on Smart Grid Communications (SmartGridComm): Data Management and Grid Analytics and Dynamic Pricing

Parallel Multi-Step Ahead Power Demand Forecasting through NAR Neural Networks

Riccardo Bonetto, Michele Rossi
Department of Information Engineering (DEI)
University of Padova, Via G. Gradenigo 6/B, 35131 Padova (PD), Italy
{bonettor, rossi}@dei.unipd.it

***Abstract*—We consider the problem of power demand forecasting in residential micro-grids. Previous approaches rely on ARMA models and, recently, on neural network architectures that are however solely used to perform one-step ahead predictions. Here, we propose an original forecasting technique based on non-linear, autoregressive (NAR) neural networks. Our architecture allows for parallel and efficient training and is also lightweight at runtime. Time series from real power demand traces are used to validate our technique, assessing its superiority with respect to state-of-the-art ARMA and ARIMA estimators. Besides smaller prediction errors in the mean and the variance, the proposed NAR architecture provides reliable indications of rises in the power demand even when predictions are generated for a time span of 2 hours. In this case, standard ARMA and ARIMA models entirely fail.**

I. Introduction

A great amount of work has been carried out to boost the efficiency of electrical grids in the presence of end users with power generation capabilities (prosumers), see for instance [1]–[3]. Demand response policies that allow prosumers to efficiently set their electrical consumption behavior with respect to economic and power efficiency benefits are being developed [4], [5]. Moreover, systems providing the prosumers with the ability of directly trading their excess energy have been devised [6].

These techniques entail optimization policies involving the knowledge of current power demand and energy availability from renewable sources. Also, further benefits can be attained through the prediction, up to a certain accuracy, of the power consumption from the loads within future time windows, as this allows using multiple-step-ahead control algorithms. For example, smart policies can be applied to assess what fraction of the generated power has to be locally stored for later use and what fraction of it can instead be fed to the loads or injected into the grid [7], [8]. To determine the future power demand, a careful planning of the daily home activities involving the use of electrical power can be thought of [9].[1] This approach, however, somewhat constraints the users to perform their activities at planned times just for the sake of the smart grid's optimization [10]. Another, perhaps more desirable approach, entails the use of historical data to forecast the power consumption within a certain future time frame.

To this end, several techniques have been developed and an increasing attention is being paid to Artificial Neural Networks (ANN) [11]. ANNs are mathematical models that allow approximating any nonlinear function [12] and, as such, can be trained to forecast power load profiles. However, standard ANNs (such as feed forward structures) are often difficult to design (i.e., in terms of size and depth) and train (due to the required high computational power).

Recent work promisingly exploited ANNs to forecast power consumption data, see for example [13]–[17]. Such work however falls short when dealing with (possibly variable) multi-step ahead forecasting entailing a long time horizon, i.e., when the output corresponds to a vector of forecast power demands into the future. The reason for that is that ANNs' topology is static. Hence, once trained to forecast W time steps into the future, the same ANN cannot be reused to forecast $W+1$ time steps, but in general it has to be redesigned and retrained. Another reason is that, as the network size increases, so does the computational power required to train it, making it infeasible for off-the-shelf computing hardware to accomplish the training task. Here, we cope with these issues by presenting an original approach that relies on a single (and small) *non linear autoregressive (NAR)* ANN [18] that at the time of deployment is trained multiple times in a parallel fashion. Each training phase produces an ANN capable of forecasting a single future value, placed i steps ahead with $i = 1, \dots, W$, where W is the prediction window. Different configurations are obtained for the same ANN structure, loading each configuration in a dedicated instance of the ANN. Each one of these ANN instances is then utilized to generate a specific element i of the forecast vector. Since no communication is required among these instances, the generation of forecast values can be performed in parallel. Moreover, should the length of the multi-step forecast vector need to be extended from W to $W+n$, only n additional network configurations have to be generated, while the previous W ones remain unchanged.

Thinking of a scenario where control policies and the corresponding forecast algorithms are trained locally at each household, our solution is found to be lightweight. Specifically, obtaining a single trained NAR instance takes on average less than 20 seconds on consumer-grade personal computers. This makes the proposed approach suited for use within prosumer's installations with off-the-shelf hardware.

The rest of this work is structured as follows. In Section II we introduce the mathematical notation used throughout the paper and provide an introductory background to ANNs and NAR networks. In Section III we discuss the use of NAR

[1]We remark that, although in this paper we refer to power consumption in households, our approach is general and can be used for any type of load.

978-1-5090-4075-9/16/$31.00 ©2016 IEEE

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.

networks for forecasting purposes and we formally present our novel parallel and lightweight forecasting technique. In Section IV we describe the experimental setup that we considered to assess its performance, and in Section V we test our NAR-based forecasting approach against ARMA and ARIMA models, which are estimated to accomplish the same task. Finally, in Section VI we draw our conclusions.

II. Mathematical Notation and Introductory Background

In this section, we introduce the mathematical notation used throughout the paper, along with some introductory backgound on artificial neural networks.

A. Mathematical Notation

Let $X = (x_1, x_2, \dots) \in \mathbb{R}^\infty$ be an infinte real valued time series. $\tau \in \mathbb{N}$ represents the current time step and $X_h^k = (x_h, \dots, x_k)$, $\forall\, h, k \in \mathbb{N}$ such that $h \leq k$, is referred to as the (h,k)-subsequence of X.

We define a η-step forecast of the subsequence X_h^τ as a finite real valued sequence $\hat{X} = (\hat{x}_1, \dots, \hat{x}_\eta) = \bar{f}_{h,\eta}(X_h^\tau) \in \mathbb{R}^\eta \;:\; \eta \in \mathbb{N} \setminus \{\infty\}$ such that $x_{\tau+i} = \hat{x}_i + e_i$, $i = 1, \dots, \eta$, where term $e_i \in \mathbb{R}$ represents the forecast error and $\bar{f}_{h,\eta} : \mathbb{R}^h \to \mathbb{R}^\eta$ is the forecast function. The parameter h determines the amount of knowledge about the past realizations of the time series that is available for forecasting.

B. Artificial Neural Networks

Artificial neural networks (ANNs) are a class of mathematical models allowing to represent, up to any accuracy, any non linear function by means of a combination of the outputs of a number of non linear, bounded and monotonically increasing functions, which are called *activation functions*. For this reason ANNs are considered universal approximators [12].

Let $\sigma : \mathbb{R} \to [\alpha, \beta] \subset \mathbb{R}$ be an activation function, then a neuron is defined as a computational unit that takes as input a real-valued sequence $\Theta = (\theta_1, \dots, \theta_n) \in \mathbb{R}^n$ and performs the transformation

$$\sigma\left(\sum_{j=1}^{n} w_j \theta_j + b\right).$$

The parameters $w_j \in \mathbb{R},\ j = 1, \dots n$ are called weights, and the parameter b is called bias.

ANNs are usually composed of many neurons organized in a cascade of layers. Each layer takes as input the output of the previous one. The first layer is usually called the input layer, as it takes as input the raw data to be processed by the ANN, while the last layer is called the output layer. An example feed forward neural network is shown in Fig. 1. This network is said to have a fully-connected structure as every neuron in layer n is connected through a different weight to any other neuron in the next layer $n+1$. Full-connectivity implies high computational complexity in the training phase.

The process by which the weights and biases are progressively tuned to produce the desired output is called learning. The learning process is performed on a set of data for which the expected network results are known, which is referred to as the training set. The performance of an ANN is measured by means of a cost function that determines a distance measure between the desired output and the actual one. Weights and biases are iteratively adjusted in order to minimize the aforementioned cost function.

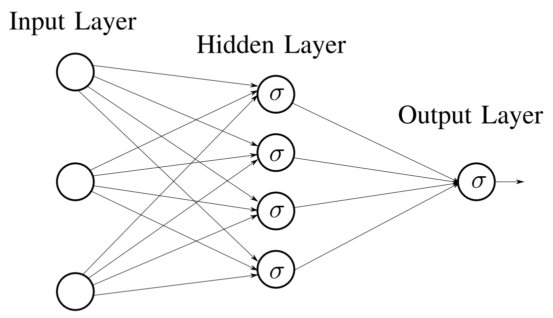

Fig. 1. Example of ANN with 3 inputs, one hidden layer with 4 neurons and one output neuron.

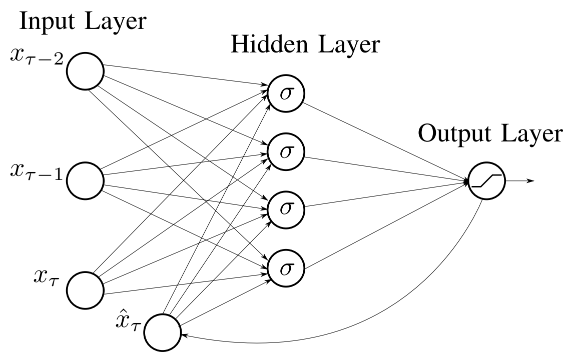

Fig. 2. Example of NAR network with 3 inputs, one hidden layer with 4 neurons and 1 output neuron.

C. Non linear AutoRegressive Neural Networks

Non linear autoregressive neural networks (NARs) are a particular class of ANNs designed to perform regression tasks on time series [18]. A NAR network operates on a time series X by processing, at each time step τ, the subsequence X_h^τ (i.e., the last h values of X) and the previous NAR's output. The parameter h determines how far in the past the network tries to capture the correlation structure of the input data sequence. The output layer is composed of neurons with linear activation functions $\mathcal{L} : \mathbb{R} \to]\alpha, \beta[\subseteq \mathbb{R}$, defined as

$$\mathcal{L}(x) = \begin{cases} x & \text{if } \alpha < x < \beta \\ \alpha & \text{if } x \leq \alpha \\ \beta & \text{if } x \geq \beta \end{cases}.$$

Fig. 2 shows an example NAR network. This network takes as input the sequence $X_{\tau-2}^\tau$ and the output generated by the network during the previous time step. These inputs are processed by a hidden layer composed of four neurons with $\sigma(\cdot)$ activation functions. The output of the hidden layer is then processed by a linear neuron to produce the desired result.

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.

III. Lightweight Parallel Multi-Step Ahead Forecasting with NAR Networks

In this section, we first introduce how NAR networks can be used for forecasting. Then, we present an original and lightweight approach to perform real time multi-step ahead forecasting with a NAR network architecture.

A. Forecasting with NAR networks

Being universal approximators with virtually unbounded output, NAR networks can also learn to forecast a time series. A forecasting function $\bar{f}^*_{h,\eta}$ such that $\bar{f}^*_{h,\eta}(X^\tau_h) = \hat{X}$ and $e_i = 0,\ i = 1, \ldots, \eta$, is said to be optimal. According to the universal approximation theorem, a NAR network can learn to approximate $\bar{f}^*_{h,\eta}(X^\tau_h)$ up to any precision, provided that it has enough neurons arranged into the right structure. Usually, NAR networks are used to perform day ahead forecasting, i.e., to forecast the value of the time series for the next time step [19], [20]. However, they can also be trained to forecast the value of the time series η time steps ahead. Moreover, NAR networks can be configured and trained to perform multi-step ahead forecasting, provided that the output layer is made of η linear neurons. While training a NAR network to forecast a single value is a relatively inexpensive task, training a NAR network to perform a multi-step ahead forecast dramatically increases the computational power required for the training process and the storage capabilities required to manage the training set.

B. Lightweight NAR networks for forecasting

Our approach to multi-step forecasting uses a single NAR network topology with different sets of weights and biases. Each of these sets is obtaining by training the network to forecast a single value a certain number of time steps ahead. In particular, consider the forecast function $\bar{f}_{h,\eta}(X^\tau_h)$, then $\mathcal{W} = \{W_1, \ldots, W_\eta\}$ and $\mathcal{B} = \{B_1, \ldots, B_\eta\}$ are the sets of weights and biases such that:

$$\bar{f}_{h,\eta}(X^\tau_h) \approx (\varphi(W_1, B_1, (X^\tau_h, \hat{x}_1)), \ldots, \varphi(W_\eta, B_\eta, (X^\tau_h, \hat{x}_\eta))),$$

where $\varphi(W_i, B_i, (X^\tau_h, \hat{x}_i)),\ i = 1, \ldots, \eta$ is the output of the considered NAR network obtained utilizing the weights W_i and the biases B_i which have been optimally calculated to forecast the value $x_{\tau+i}$ of the input time series X.

By doing so, a relatively small single output NAR network can be trained η times on the same training set, storing the corresponding weights and biases. By having a small NAR network, the storage capabilities required to manage the training set are considerably reduced with respect to larger multi output networks. Moreover, the training phases and the η-step ahead forecast can be performed in a completely parallel fashion with no communication between the network instances. These facts result in a considerable speedup of the training phase and in an extremely fast forecast computation, thus making the proposed approach suitable for small home devices even for long time windows (large η).

Fig. 3 shows an example of the proposed procedure. The subsequence X^τ_h of the input time series X is broadcast

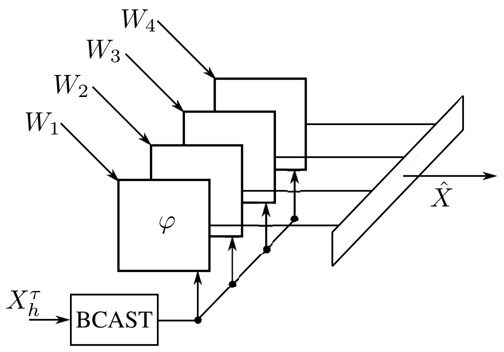

Fig. 3. Example of an ANN with 3 inputs, one hidden layer with 4 neurons and one output neuron.

(BCAST block) to the four NAR network instances represented by the white squares. Each instance $i = 1, \ldots, 4$ computes the function $\varphi(W_i, B_i, (X^\tau_h, \hat{x}_i))$, which generates the i-th forecast $\hat{x}_i$. The four forecasts are then ordered into the sequence $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_4)$ which represents the system output.

Algorithm 1 Parallel Multi-Step Ahead Forecasting

```
Input $X^\tau_h$: Time series subsequence
Input $\varphi$: NAR network Function
Input $(\mathcal{W}, \mathcal{B})$: Weight and Bias sets
Input $p$: number of CPUs
Input $L$: forecast length
for $i = 1$; $i \le L$; $i$++ do
    Send $X^\tau_h$, $W_i$, $B_i$ to CPU $(i-1)\%p + 1$
end for
Each CPU computes $\varphi$ on the received data
for $i = 1$; $i \le L$; $i$++ do
    Gather $\hat{x}_i$ from CPU $(i-1)\%p + 1$
end for
Return $\hat{X} = (\hat{x}_1, \ldots, \hat{x}_{L-1})$
```

Algorithm 1 formalizes in pseudocode the parallel procedure described so far. After receiving as input the neural network function, the weight and bias set, the current time series subsequence and the desired forecast time span, the algorithm cyclically distributes the inputs to the available CPUs. These, in turn, compute $\varphi(\cdot)$ according to the data they receive. The single forecasts are finally collected into the output vector $\hat{X}$.

IV. Experimental Setup

Here, we present the experimental setup that we used to assess the performance of the proposed approach. Taking inspiration from [3], and considering that many smart grid application (as, for example, demand response policies [4], [5] and peer to peer energy trading [6]) are expected to obtain a considerable benefit through an accurate forecast of the power demand, we considered the aggregated power consumption of a single household [21]. We focused on the time series containing active power measurements taken every 60 seconds during a period of 4 years.

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.

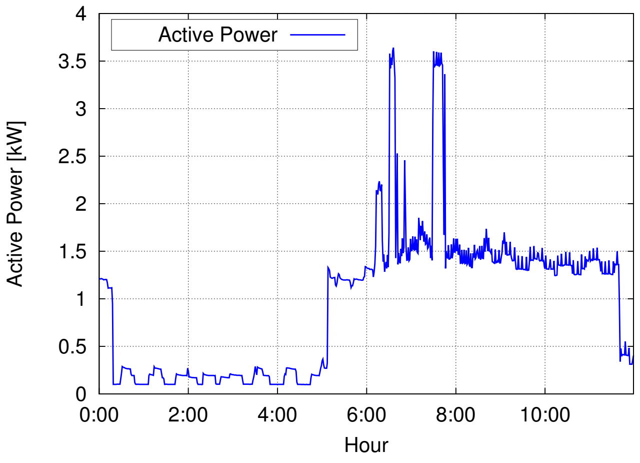

Fig. 4. 12 hours active power consumption for a single household. France, December 2006.

Fig. 4 shows an example power demand time series. Two main behaviors appear during a time span of twelve hours. In the first hours of the day, the power consumption is fairly low and presents an almost cyclic pattern. Starting from five o' clock in the morning, the power consumption suddenly increases, probably due to the heating system kicking in as the data refers to France, in December 2006. The power consumption then drops again around 11:30 AM.

The NAR network that we used for the results in the next section is configured as follows:

- it takes the subsequence $X_{\tau-29}^{\tau}$ as input (i.e., a time horizon of 30 minutes is used to forecast future values);
- it has one hidden layer with 40 neurons, each of them with activation function $\sigma(x) = \frac{2}{(1+e^{-2x})} - 1$;
- it has one output neuron with linear activation function $\mathcal{L}(x) = x$.

The network has been trained using the first $5,000$ samples of the considered time series, and the weight and bias sets have been obtained for forecasts spanning from 1 to 120 minutes ahead of the current time step.

The chosen training algorithm is the Levenberg-Marquardt with Bayesian weights regularization [22]. This algorithm is particularly suited for time series exhibiting a noisy behavior. For this reason, it is expected to accurately track the sudden drops and rises in the active power demand (see Fig. 4 for an illustrative example). The performance measure with respect to which the weights and biases are adjusted during the ANN training is the mean square error between the network outputs and the forecast targets. 120 instances of this network have been generated and each of these has been trained to predict one single forecast value (from 1 to 120 minutes ahead) for 25 training epochs. The forecast performance has then been tested on the portion of the time series that was not used for training purposes (approximately 2×10^6 samples).

The performance of the proposed forecasting technique has been assessed in terms of mean absolute error and error variance of the predicted values with respect to the actual ones. The obtained data have then be compared to the ones obtained by an ARMA model with 30 regressors and Gaussian noise, and an ARIMA model with 30 regressors, Gaussian noise and first order differentiation. These models have been estimated using the same training set as that considered for the NAR network [23].

V. Results

In this section, we show some selected numerical results obtained from the models of Section IV. First, in Section V-A we present a qualitative comparison between the forecast vectors obtained through the proposed approach and those obtained through the ARMA model described in Section IV. Then, in Section V-B we compare the performance of the proposed NAR ANN against that of the ARMA and ARIMA models in terms of mean absolute error and error variance. Finally, in Section V-C we discuss the execution times needed to train the ANN models and to use them.

A. Qualitative Comparison

Fig. 5 qualitatively shows the performance of the ARMA model when used to predict the active power consumption 1, 10, 60, and 120 steps (i.e., minutes) ahead. When forecasting one step ahead, it can be noticed that the ARMA model's predictions are very close to the targets until the consumed power suddenly rises. After that, the model basically fails, as can be seen by the horizontal shift between the actual time series and the predicted output. In addition, as the forecast time span increases, the performance drastically worsens, up to the point that no useful output is produced when trying to forecast 60, and 120 steps ahead (here, the predicted power demand remains flat, although the actual time series exhibits major variations).

Fig. 6 shows the performance of the proposed ANN model when used to predict the active power consumption with the same time spans as the ones in Fig. 5. When forecasting one step ahead (one minute), the NAR ANN returns very accurate predictions, which can be seen from the fact that there is no horizontal shift between the actual time series and the predicted one. A better performance with respect to ARMA is likewise achieved for longer time spans.

We note that, although the performance of the NAR is impacted for long prediction windows (e.g., 120 minutes ahead), with this network we can actually foresee that there will be a consistent rise in the power demand at a distance of 2 hours. This is instead impossible using an ARMA model.

B. Performance Analysis

Fig. 7 shows the comparison between the mean absolute error of the forecast obtained by the ARMA and ARIMA models and that obtained with the proposed NAR ANN-based technique. The mean absolute error has been computed for each forecast as the arithmetic average of the distance between the points generated by the two models and the target values of the input time series.

The conclusions that were intuitively drawn from the results in Section V-A are confirmed here. It can be noticed

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.

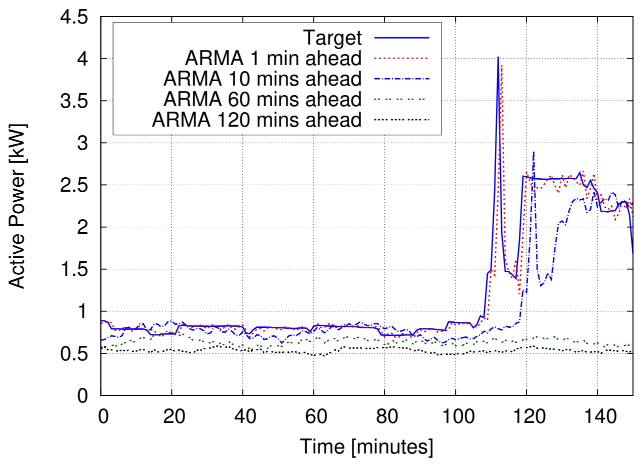

Fig. 5. Forecasts obtained using the estimated ARMA model for 1, 10, 60, and 120 steps ahead. The step size is one minute.

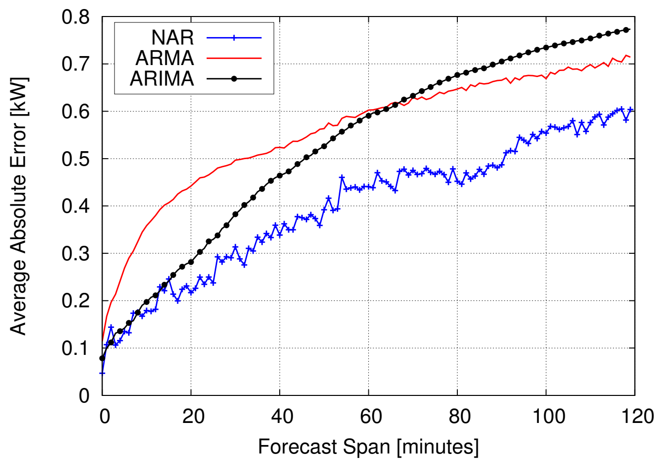

Fig. 7. Mean absolute error for the estimated ARMA, ARIMA models and the proposed NAR ANN-based forecasting scheme.

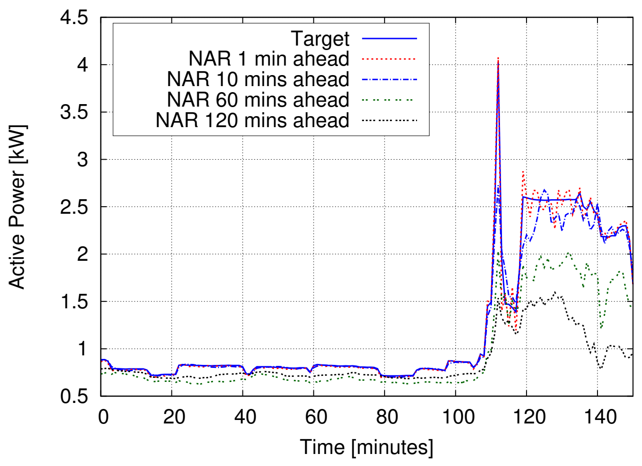

Fig. 6. Forecasts obtained using the trained NAR network for 1, 10, 60, and 120 steps ahead. The step size is one minute.

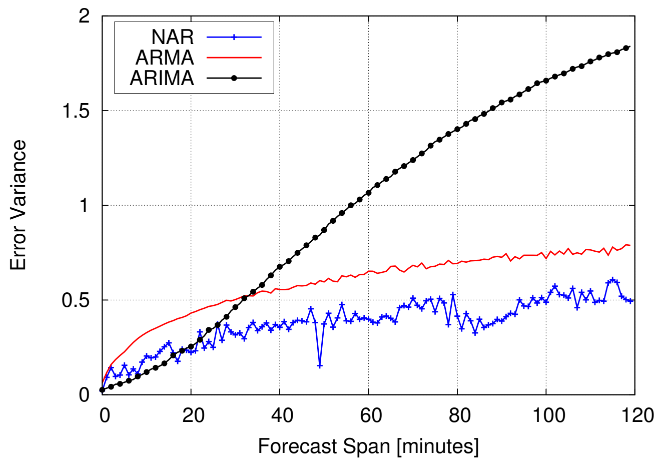

Fig. 8. Error variance for the estimated ARMA, ARIMA models and the proposed NAR-ANN based forecasting scheme.

that for the first steps ahead the forecasts generated by the two models are quite close, while, as the time span of the forecast increases, our NAR-ANN based framework generates considerably better results in terms of mean absolute error. Moreover, the error for the neural network model grows almost linearly, whereas that of ARMA exhibits a much faster growth, especially in the first 20 points. As for the ARIMA model, for forecast periods within 60 minutes, the prediction error is in between that of ARMA and NAR, but the error variance grows much faster than the one of the other two models.

In Fig. 8, we compare the variance of the prediction errors of Fig. 7. As a first result, we note that the error variances exhibit the same growth trend of the corresponding errors of Fig. 7. As expected, this confirms that an increasing time window corresponds to an increasing uncertainty in the prediction accuracy. Nevertheless, especially in the first 20 points the uncertainty related to the error of the ARMA model grows much faster than that of the proposed NAR ANN-based approach, making the latter more resilient to the length of the prediction window. Another noticeable result is that the proposed approach is always more accurate than the ARMA model while, at the same time, granting a higher confidence on the forecast data. With respect to the ARIMA model, it can be noticed that the corresponding error variance is the smallest in the first 20 points. However, beyond these, the uncertainty of the forecasts grows much faster than that of ARMA and NAR.

According to the results of Fig. 8 and Fig. 7, the proposed approach shows the best results for forecasts that span more than 20 steps-ahead. During the first 20 time steps, instead, the results of the ARIMA model are very close to those obtained by the proposed technique. In particular, for these the ARIMA model guarantees a slightly higher error, but with a slightly smaller uncertainty.

C. Experimental Computational Efficiency

The Achille's heel of neural networks is the computational and storage power required to train them. The presented

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.

Forecast Window [minutes]	Computational Time [s]
1	2
10	10
60	130
120	241

TABLE I
COMPUTATIONAL TIME REQUIRED TO PERFORM MULTI-STEP AHEAD FORECAST THROUGH AN ARMA MODEL.

approach makes it possible to train multiple times a small network in a parallel fashion and to outperform the results generated by ARMA and ARIMA models. The estimation of the NAR model parameters only took 13 seconds on a quad-core i7 Intel CPU. Being able to exploit the parallelism of the i7 quad-core architecture, the total training time of the neural network to obtain a 120-step ahead forecast was approximately 8 minutes and 30 seconds. Once the training phase is completed, each forecast requires from 4 to 5 seconds to be executed. Using the ARMA model, the time required to perform the computation increases as the time span of the forecast grows, as shown in Tab. I. Similar results have been obtained for the ARIMA model.

VI. Conclusions

In this work we have presented a novel and effective, NAR ANN-based, parallel technique for multi-step time series forecasting. After formally introducing the NAR ANNs, we have devised a general formulation of a lightweight and parallel forecasting algorithm based on these networks. We have then focused on the prediction of active power consumption data for a single household and we have assessed the performance of the proposed algorithm with respect to that obtained using state-of-the-art ARMA and ARIMA models. Experimental results show that our approach outperforms previous forecasting solutions in terms of prediction error and error variance for long-term forecasts, looking as far as 120 steps (minutes) ahead into the future. Moreover, we qualitatively analyzed the computational power required by our approach, showing that it is well suited for off-the-shelf computing hardware. Owing to the good performance in terms of prediction accuracy and reliability, and to its moderate computational demand, the proposed forecasting technique can be utilized by prosumers as a building block of future smart grid infrastructures.

Acknowledgements

The work in this paper has been supported by the University of Padova through the Junior Research Grant ENGINE: "ENGineering the INternet of Energy".

References

[1] P. Tenti, A. Costabeber, P. Mattavelli, and D. Trombetti, "Distribution loss minimization by token ring control of power electronic interfaces in residential microgrids," *IEEE Trans. Industrial Electronics*, vol. 59, no. 10, pp. 3817–3826, Oct. 2012.
[2] R. Bonetto, M. Rossi, S. Tomasin, and M. Zorzi, "On the interplay of distributed power loss reduction and communication in low voltage microgrids," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 322–337, Feb 2016.
[3] R. Bonetto, T. Caldognetto, S. Buso, M. Rossi, S. Tomasin, and P. Tenti, "Lightweight energy management of islanded operated microgrids for prosumer communities," in *IEEE International Conference on Industrial Technology (ICIT)*, Seville, Spain, March 2015, pp. 1323–1328.
[4] P. B. Luh, L. D. Michel, P. Friedland, C. Guan, and Y. Wang, "Load forecasting and demand response," in *IEEE PES General Meeting*, Minneapolis, MN, U.S., July 2010.
[5] I. Dusparic, C. Harris, A. Marinescu, V. Cahill, and S. Clarke, "Multi-agent residential demand response based on load forecasting," in *IEEE Conference on Technologies for Sustainability (SusTech)*, Portland, OR, U.S., Aug 2013.
[6] Y. Luo, S. Itaya, S. Nakamura, and P. Davis, "Autonomous cooperative energy trading between prosumers for microgrid systems," in *Local Computer Networks Workshops (LCN Workshops), 2014 IEEE 39th Conference on*, Edmonton, AB, CA, Sept 2014.
[7] B. Narayanaswamy, T. S. Jayram, and V. N. Yoong, "Hedging strategies for renewable resource integration and uncertainty management in the smart grid," in *IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Berlin,DE, Oct 2012.
[8] R. Haque, T. Jamal, M. N. I. Maruf, S. Ferdous, and S. F. H. Priya, "Smart management of phev and renewable energy sources for grid peak demand energy supply," in *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*, Dhaka, BD, May.
[9] C. Eksin, H. Deli, and A. Ribeiro, "Distributed demand side management of heterogeneous rational consumers in smart grids with renewable sources," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, IT, May 2014.
[10] T. Logenthiran, D. Srinivasan, and T. Z. Shun, "Demand side management in smart grid using heuristic optimization," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1244–1252, Sept 2012.
[11] G. K. Venayagamoorthy, "Potentials and promises of computational intelligence for smart grids," in *2009 IEEE Power Energy Society General Meeting*, Calgary, CA, July 2009.
[12] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.
[13] P. Qingle and Z. Min, "Very short-term load forecasting based on neural network and rough set," in *IEEE International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Changsha, CN, May 2010.
[14] Y. Ning, Y. Liu, H. Zhang, and Q. Ji, "Comparison of different bp neural network models for short-term load forecasting," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, Xiamen, CN, Oct 2010.
[15] D. Niu, H. Shi, J. Li, and Y. Wei, "Research on short-term power load time series forecasting model based on bp neural network," in *IEEE International Conference on Advanced Computer Control (ICACC)*, Shenyang, CN, March 2010.
[16] W. Dai and P. Wang, "Application of pattern recognition and artificial neural network to load forecasting in electric power system," in *IEEE Third International Conference on Natural Computation (ICNC 2007)*, Haikou, CN, Aug 2007.
[17] S. l. Liu, Z. q. Hu, and X. k. Chi, "Power load forecasting based on neural network and time series," in *IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, Beijing, CN, Sept 2009.
[18] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
[19] K. Gairaa, F. Chellali, S. Benkaciali, Y. Messlem, and K. Abdallah, "Daily global solar radiation forecasting over a desert area using nar neural networks comparison with conventional methods," in *International Conference on Renewable Energy Research and Applications (ICRERA)*, Palermo, IT, Nov 2015.
[20] K. A. Baharin, H. A. Rahman, M. Y. Hassan, and G. C. Kim, "Hourly irradiance forecasting for peninsular malaysia using dynamic neural network with preprocessed data," in *Research and Development (SCOReD), 2013 IEEE Student Conference on*, Putrajaya, MY, Dec 2013.
[21] K. Bache and M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[22] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995.
[23] C. Chatfield, *The Analysis of Time Series: An Introduction. Sixth Edition.* Florida, U.S.: CRC Press, 2016.

Authorized licensed use limited to: University of New Brunswick. Downloaded on June 15,2020 at 21:06:47 UTC from IEEE Xplore. Restrictions apply.