2023 10th International Conference on Power and Energy Systems Engineering (CPESE) | 979-8-3503-2762-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/CPESE59653.2023.10303240

Emission Mitigation Dispatch in Wind Power Generation with Expedited Machine Learning

Xian Liu*
Department of Systems Engineering
University of Arkansas at Little Rock
Little Rock, USA
xxliu@ualr.edu
*Corresponding author

Changcheng Huang
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
huang@sce.carleton.ca

***Abstract*—In this paper: a machine learning (ML) module is presented for emission mitigation dispatch (EMD) in wind power (WP) generation. It is developed for the mobile edge computing (MEC) platforms. Therefore, the module is based on a lightweight ML scheme: radial basis neural network (RBN). It has been well known that RBN is compact compared with a more popular ML scheme: feedforward neural network. However, this is the first study to combine EMD with RBN, strongly motivated by the MEC implementations. MEC is becoming an important platform in the evolving smart grid program. As a trial, a case study with 36 generators and 50 wind turbines is presented. Two standard ML stages, training and testing, are conducted in the simulation. It is shown that, by using the trained RBN, the time of solving EMD is significantly reduced. It is expected to be a desirable model for MEC.**

***Index Terms*— Artificial intelligence, emission mitigation, machine learning, electric smart grid, wind power.**

I. INTRODUCTION

One of the most prominent features of *electricity smart grid* (ESG) is integrating modern communication facilities, primarily the Internet, into the power grid. However, there are several distinct features in communications over ESG. For example, a significant portion of communication tasks is control-oriented. Examples include decision-making communications for optimizing energy utilization and such. This is especially important when renewable energy supplies are incorporated, such as wind power generation. This is because wind is highly dynamic in nature and the decisions must be promptly made for electricity scheduling and delivery. *Machine learning* (ML) would play a very active role in enhancing the decision-making process.

Modern power dispatch typically goes through three segments: *generation* (from fossil fuels), *transmission*, and *distribution* (commonly referred to as GTD in the utility industry). In the area of communications, there was a strong interest in ESG's *neighborhood area networks* (NAN) ([2], [3]). In practice, a large portion of green energy generating utilities would be deployed near the segment of NANs. Therefore, the end devices will play an increasingly important role. Currently there is a growing consensus that the ML should be expeditiously introduced to the front end of smart grid communications. In fact, the interest in *mobile edge leaning* (MEL) is high due to the latest progress of *beyond-the-fifth generation* (B5G) technologies [4].

In this paper, we describe a lightweight ML module for *emission mitigation dispatch* (EMD) incorporating WP. Since the learning is to be done in the end devices, where the computing capacity is limited, we seeks small scale ML approaches. Our study found that the *radial basis neural network* (RBN) strategy performs much faster than *feedforward neural network* (FNN). Therefore, it is feasible to implement this lightweight ML module into mobile devices to exert the MEL's ascendancy. It is expected that the end platform of ESG can run this MEL client to find the locally optimal solution for scheduling WP. This solution is geographically local. Then these local solutions can be forwarded to a control hub to form an overall solution for the concerned area.

The rest of this paper is organized as follows. In Section II, a generic EMD model is reviewed, and the WP behavior is described. Then, in Section III, the essentials of RBN are highlighted. Next, in Section IV, the details of the training process and testing process are presented. Finally, conclusions are included in Section V.

II. BACKGROUND

In electrical power systems, emission mitigation dispatch (EMD) is a variant of *economic load dispatch* (ELD) [5]. The main objective of EMD is to reduce the impacts caused by gaseous pollutants such as *carbon oxides*, *oxides of nitrogen* (NOx), and *sulfur oxides* (SOx) [6]. EMD is naturally a constrained nonlinear optimization problem. Recent studies on EMD usually take renewable energy into account [7]. When the *wind power* (WP)[1] is included in the supply-demand constraint, EMD becomes a stochastic optimization problem. Since WP is highly volatile, the EMD problem needs to be solved quickly. This is especially important for the management of short-term WP. The short-term is commonly agreed as ranging from 1 hour up to 72 hours. However, most

[1] Throughout this paper, WP is referred to as the electric power generated by wind turbines, i.e., *wind power generation* (WPG).

conventional solving methods are iteration-based and time-consuming. In this paper, we apply the *machine learning* (ML) methodology to solve EMD with short-term WP.

Nowadays, the ML methodology plays a primary role in *artificial intelligence* (AI) [8]. In general, AI explores the capability of thinking and learning by computer systems. ML is typically built upon the *artificial neural networks* (ANNs). Up to date, a great deal of studies has revealed that ANN can improve the solving efficiency for a wide range of optimization problems. There are several types of ANN. The representatives include the *feedforward neural network* (FNN, a.k.a. *multilayer perceptrons*, MLP), *convolutional neural network* (CNN), *recurrent neural network* (RNN), and *radial basis neural network* (RBN), to name a few. For some specific optimization problems, it is highly beneficial to know which type of ANN is more effective and efficient than others. In the present work, we investigate the performance of RBN applied to solve the NOx problem with WP.

RBN has received an intensive interest since the early era of ML discipline [9]. The advantages of RBN include almost all desirable features of a good ANN: universal approximator, simple architecture, efficient training procedure, and effective generalization capacity. Similar to FNN, RBN can approximate a wide range of functions. However, RBN has only a single hidden layer, hence a very compact structure and the rapid training process. Moreover, RBN is robust against the noise in input dataset. On the other hand, a main drawback of RBN is the possibly large number of neurons, mainly due to its single-layer structure. However, for the small-scale through mid-scale applications, the size of hidden layer can be well controlled. Overall, the main features of RBN make it a promising facility to solve a wide range of optimization problems in real-time mode. In this study, we show that RBN can quickly solve the EMD problem with WP.

Currently, electric power is mainly generated by the thermal turbines fueled with coal, oil, or gas. Therefore, the power generation produces gaseous pollutants such as carbon oxides, NOx, and SOx [6]. As a case study, the present work chooses the emission produced by NOx as the primary concern. However, with some refinements, the modeling and solving schemes can be easily applied to other types of emissions.

In the literature, the following expression has been commonly used to characterize the impact of NOx emission [5, 7, 10, 11]:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 \exp(a_4 x), \tag{1}$$

where a_k $(k = 0,1,\cdots,4)$ are the coefficients estimated from experiments, y is the *environmental impact index* (EII), with the unit of kg/hour, while x is in the *per-unit* (p. u.) value with the base 100 MVA. In the present study, we consider n thermal generators and m wind turbines. Accordingly, the EMD problem can be formatted as follows:

(EMD)

$$\text{minimize } y = \sum_{k=1}^{n} [a_{k0} + a_{k1} x_k + a_{k2} x_k^2 + a_{k3} \exp(a_{k4} x_k)]; \tag{2}$$

subject to

$$\begin{cases} \sum_{k=1}^{n} x_k = P_d + P_L, \\ x_{\min,k} \le x_k \le x_{\max,k}. \qquad (k = 1,2,\cdots,n) \end{cases} \tag{3}$$

When the wind power is considered, we have:
(EMD_WP)

$$\text{minimize } y = \sum_{k=1}^{n} [a_{k0} + a_{k1} x_k + a_{k2} x_k^2 + a_{k3} \exp(a_{k4} x_k)]; \tag{4}$$

subject to

$$\begin{cases} \sum_{j=1}^{m} w_j + \sum_{k=1}^{n} x_k = P_d + P_L, \\ x_{\min,k} \le x_k \le x_{\max,k}. \qquad (k = 1,2,\cdots,n) \end{cases} \tag{5}$$

Note that EMD_WP is an optimization problem in which y is the objective function and x_k is the decision variable. Moreover, w_j is treated as a *random variable* (RV). Some discussions would be helpful. WP is associated with wind dynamics, characterized by wind speed, altitude, atmospheric pressure, temperature, etc. In nature, these parameters fluctuate all the time in a stochastic manner and are commonly described by some RVs. In practice, the randomness of WP may be mitigated through several approaches, such as curtailment for generation and/or load. The present work mainly considers the raw effect of varying wind speed, without explicitly involving the curtailment-like schemes. In the short-term horizons like in hours, it has been shown that the WP approximately follows the *beta* distribution [12-14]. The PDF of beta RV is [15]:

$$f_Z(z) = \begin{cases} \dfrac{1}{B(a,b)} z^{a-1}(1-z)^{b-1}, & (0 < z < 1) \\ 0, & (\text{otherwise}) \end{cases} \tag{6}$$

where $a > 0, b > 0,$ and $B(a,b)$ is the *beta function* (BF):

$$B(a,b) = \int_0^1 u^{a-1}(1-u)^{b-1} du. \quad (a > 0, b > 0) \tag{7}$$

Two sets of practical data are listed in Table I, which are based on the data reported in [14, Figure 5].

TABLE I. EXAMPLES OF PARAMETERS

	a	b	$E(Z)$	var(Z)
Curve 1	9.4000	6.2667	0.6000	0.0144
Curve 2	0.6457	0.4304	0.6000	0.1156

In above formulas, the random variable Z represents the normalized WP, defined as:

$$Z = w / w_r. \tag{8}$$

For the short-term ELD problem involving multiple wind turbines ($m>1$), it is difficult to have the analytical approach. For example, even for the case of $m=2$, the PDF of WP is already complicated:

$$f_Z(z)=\left(\frac{\Gamma(a+b)}{\Gamma(b)}\right)^2\frac{z^{2a-1}(1-z)^{b-1}}{\Gamma(2a)} \times F_1\left(a,1-b,1-b,2a;\frac{z}{z-1},z\right), \quad (0<z<1) \tag{9}$$

$$f_Z(z)=\frac{B(a+b-1,a+b-1)}{B^2(a,b)}, \quad (z=1) \tag{10}$$

$$f_Z(z)=\left(\frac{\Gamma(a+b)}{\Gamma(a)}\right)^2\frac{(z-1)^{a-1}(2-z)^{2b-1}}{\Gamma(2b)} \times F_1\left(b,1-a,1-a,2b;\frac{z-2}{z-1},2-z\right), \quad (1<z<2) \tag{11}$$

where $F_1(\bullet)$ is the *Appell's first hypergeometric function of two variables*. The derivation of (9) through (11) is due to [16, (3.211)]. On the other hand, the *moment generating function* (MGF) of $m=2$ is also complicated:

$$M_w(s)=[{}_1F_1(a;a+b;s)]^2 =\left\{1+\sum_{k=1}^{\infty}\left(\prod_{j=0}^{k-1}\left(\frac{a+j}{a+b+j}\right)\right)\frac{s^k}{k!}\right\}^2, \tag{12}$$

where ${}_1F_1(\bullet;\bullet;\bullet)$ is the *confluent hypergeometric function* [16, (9.210.1)]. The derivation of (12) is due to [16, (3.383.1)]. It is more challenging to derive the *cumulative distribution function* (CDF) of the summed beta RVs. Obviously, for arbitrary case such that $m>2$, the analytical approach is intractable. Therefore, it is imperative to seek some new approaches. A promising approach is to introduce RBN.

III. ML AND RBN

The modern ML discipline is built upon the artificial neural networks (ANNs). RBN is one of the representatives of ANN. The key insight can be gained by understanding the intrinsic principle of ANN. First, consider a generic problem:

$$\begin{aligned}&\text{minimize} \quad f(x,u),\\ &\text{subject to} \quad g(x,u)\le 0,\end{aligned} \tag{13}$$

where $f:R^n\to R$, $g:R^n\to R^m$, x is the vector of decision variables, and u is the vector of parameters. In concept, the optimal point of (13) must satisfy the *Karush–Kuhn–Tucker* (KKT) conditions [17], which is generally a system of nonlinear equations:

$$\begin{cases}\nabla f(x,u)+\lambda^T\nabla g(x,u)=0,\\ \lambda^T g(x,u)=0,\end{cases} \tag{14}$$

where λ is the vector of Lagrangian multipliers, and the superscript T stands for transpose. Denote the optimal solution as x^*. Since it is influenced by u, we formally write it as:

$$x^*=h(u), \tag{15}$$

where $h(\bullet)$ is a unknown function. Next, consider the ANN methodology. In general, ANN can perform the *supervised training* or *unsupervised training* [8, 9]. ANN can be applied to *pattern classification* or *function approximation*. For the present problem, we mainly address the aspects of ANN for function approximation and supervised training. In this context, the ANN discipline commonly begins from the regression scheme.

To simplify the concept, let us consider the linear regression. Prior to describing RBN, it is instructive to briefly review the feedforward neural network (FNN). Typically, FNN consists of an input layer, multiple hidden layers, and an output layer. The dimensions of the input layer and output layer are determined by the provided training data. Two types of datasets are needed for the supervised training. Before training FNN, a standard *nonlinear programming* (NLP) solver is used to obtain a set of solutions represented by (15). Note that (15) is conditional on the parameter u. Each solution is represented by an M-dimensional vector. The target dataset consists of Q such vectors, where Q is the number of samples. The use of target dataset is one of the main features in the supervised training. The second type of dataset for FNN is commonly formatted as an $R\times Q$ matrix [18], where R is the dimension of a single input vector. Corresponding the NLP problem, R is just the number of total random parameters, while Q is the number of total scenarios. Accordingly, the dimension of input layer and is determined by R, while the dimension of output layer is determined by the target vector.

Unlike the input layer or output layer, the dimensions of hidden layers in FNN are not determined by the external datasets. Typically, a hidden layer consists of many operational units, referred to as *neurons*. These neurons are typically implemented by various *activation functions* to approximate the unknown function in (15), expressed as follows:

$$x^*=\sigma(w^Tu+c), \tag{16}$$

where w and c are commonly called *weight* and *bias*. Note that the term w^Tu is the dot product of two corresponding vectors. One of the most popular activation functions is the *logistic sigmoid*:

$$\sigma(t)=\frac{1}{1+\exp(-t)}. \tag{17}$$

The training goal of FNN is to find the optimal values of (w,c) to minimize the difference between (15) and (16). After that, for any new values of u, the formula (16) can be used to calculate the corresponding x^*. It should be

mentioned that the KKT conditions are only of theoretical significance and have a limited usage in practical optimization. Here it is just used to describe the inherent relations. Also note that the result (16) generated by FNN provides the approximate solution for each particularly realized u. Note that FNN tries to find the solution to be close to all scenarios, rather than perfectly interpolating each individual target represented by (15). RBN has several distinct features than FNN. Note that, in FNN, the argument of activation function is the dot product as shown in (16). In contrast, RBN uses the following argument in its activation functions:

$$c\|w-u\|, \tag{18}$$

where $\|\bullet\|$ is the Euclidian distance. One of the most popular activation functions in RBN is the Gauss function:

$$\sigma(t) = \exp(-t^2). \tag{19}$$

There are several types of RBN. Two of the most representative implementations in Matlab are `newrbe` and `newrb`. The former is mainly used for exact function interpolations [19], while the latter is often used for function approximation. Since the ultimate mission of ML is to process the data not seen in the training phase, the exact interpolation is usually not a desired scheme. One of the main reasons is to mitigate overfitting [8, 9].

IV. SIMULATIONS

A. Power System Setup

The main system parameters are presented in Table II. In Tables III and IV, the coefficients of six generators are listed as examples. These data were used in previous studies on EII reduction (e.g., [5]). The transmission loss is omitted for simplicity. Note that, in columns 1 through 3 of Table III, "10^{-2}" means that every datum should be multiplied by 10^{-2}. For example, the actual value of a_{10} is 0.04091.

TABLE II. MAIN PARAMETERS IN TESTBED

Parameter	Value
Number of thermal generators	36
Number of wind turbines	50
Rated WP of wind turbine	0.048 (p.u.)
Total load demand	24 (p.u.)
beta (a,b) (1-hr horizon)	$(9.4000, 6.2667)$
beta (a,b) (48-hr horizon)	$(0.6457, 0.4304)$

TABLE III. NOX EMISSION COEFFICIENTS OF THERMAL GENERATORS

Generator Index i	a_{i0} (10^{-2})	a_{i1} (10^{-2})	a_{i2} (10^{-2})	a_{i3}	a_{i4}
1	4.091	-5.554	6.490	2e-4	2.857
2	2.543	-6.047	5.638	5e-4	3.333
3	4.258	-5.094	4.586	1e-6	8.000
4	5.326	-3.550	3.380	2e-3	2.000
5	4.258	-5.094	4.586	1e-6	8.000
6	6.131	-5.555	5.151	1e-5	6.667

TABLE IV. OPERATION RANGE OF THERMAL GENERATORS

Generator Index i	$x_{\min,i}$ (p.u.)	$x_{\max,i}$ (p.u.)
1	0.02	0.5
2	0.03	0.6
3	0.05	1.0
4	0.06	1.2
5	0.05	1.0
6	0.03	0.6

B. Training by RBN

In this study, we solve the ELD with WP by a radial basis neural network (RBN). The training and testing are conducted with `newrb` in Matlab [18]. Main parameters are in Table V.

TABLE V. MAIN PARAMETERS IN RBN

Parameters	Value
`Maximum number of neurons`	Q
`Spread`	1
`MSE goal`	0

In Table V, Q is the total number of input samples. In this simulation, Q = 274 was used. Each sample is a point in the R-dimensional space. In other words, each sample is a vector consisting of R components. These samples are generated by the NLP solver `fmincon` in Matlab. Three instances of the targets (one-hour horizon and 48-hour horizon) are presented in Tables VI and VII, respectively. The training procedure converged at $\text{MSE} \approx 1.419\times10^{-13}$ and 1.028×10^{-9}, respectively for 1-hour and 48-hour horizons (Figs. 1 and 2).

TABLE VI. EXAMPLES OF TARGET MATRIX (ONE-HOUR HORIZON)

Row	Column 1	Column 100	Column 200
1	0.5000	0.5000	0.5000
2	0.5942	0.5949	0.5959
…	…	…	…
35	0.7284	0.7294	0.7308
36	0.6000	0.6000	0.6000

TABLE VII. EXAMPLES OF TARGET MATRIX (48-HOUR HORIZON)

Row	Column 1	Column 100	Column 200
1	0.5000	0.5000	0.5000
2	0.6000	0.5947	0.5938
…	…	…	…

35	0.7388	0.7291	0.7279
36	0.6000	0.6000	0.6000

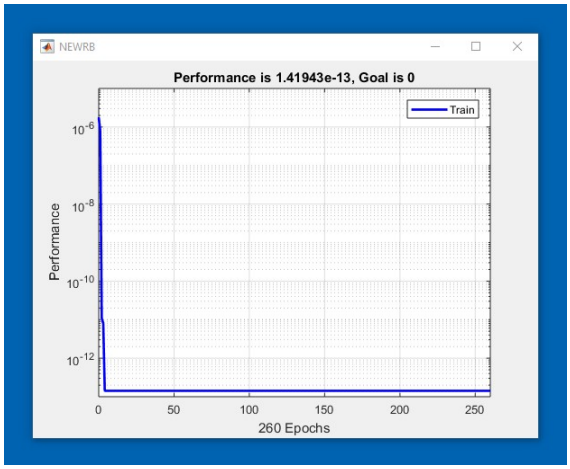

Figure 1. Training performance with the dataset of one-hour horizon.

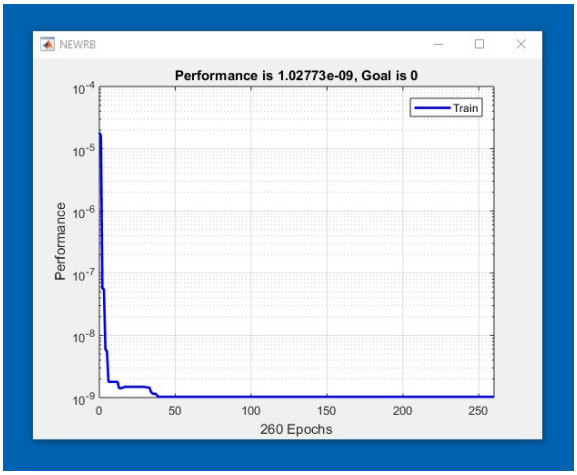

Figure 2. Training performance with the dataset of 48-hour horizon.

C. *Testing by RBN for One-Hour Horizon*

To test the obtained RBN, the dataset of the one-hour horizon WP scenario is firstly generated by the NLP solver `fmincon`. Note that this dataset is different than the data used for training. This newly generated dataset is denoted as the "reference" data in the following. Then, the RBN is used to conduct testing. There are several metrics to evaluate the performance. Here we choose the sum of power produced by all thermal generators. Note that this metric is equivalent to the 1-*norm* of the solution vector. In Fig. 3, this sum is plotted against the random wind power. The result of RBN is compared with the reference data. In Fig. 4, the relation between the RBN output and the reference is illustrated. The testing results for 48-hour horizon are illustrated in Figs. 5 and 6. It is observed that RBN has shown a very good matching.

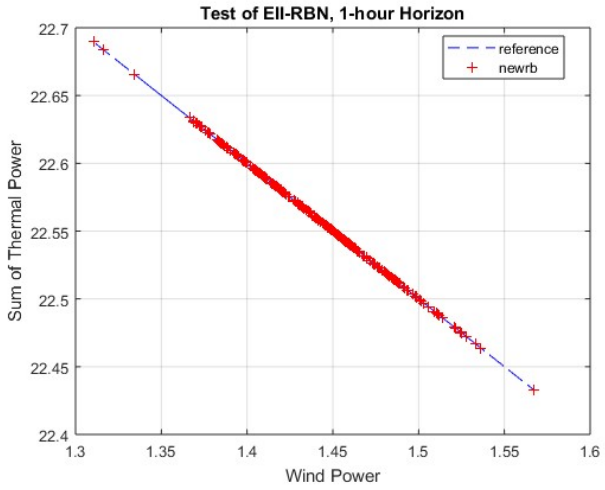

Figure 3. Testing for one-hour horizon, thermal power vs. WP.

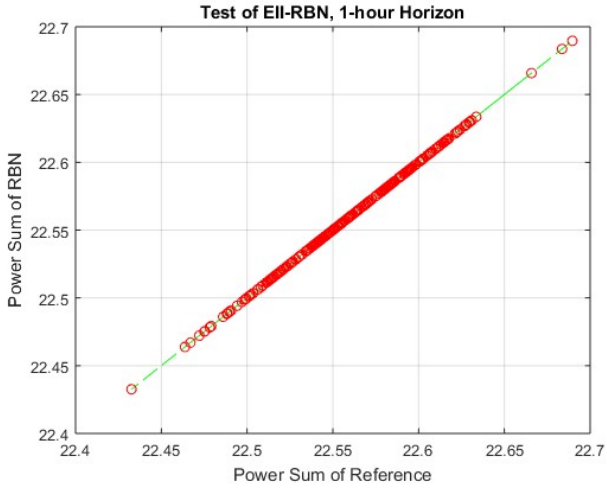

Figure 4. Testing for one-hour horizon, RBN vs. reference.

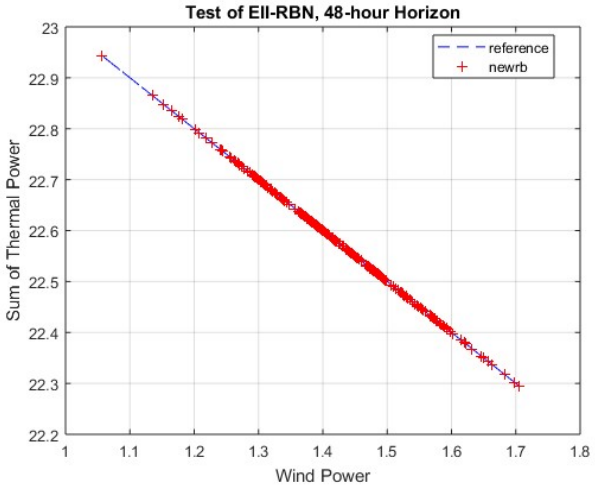

Figure 5. Testing for 48-hour horizon, thermal power vs. WP.

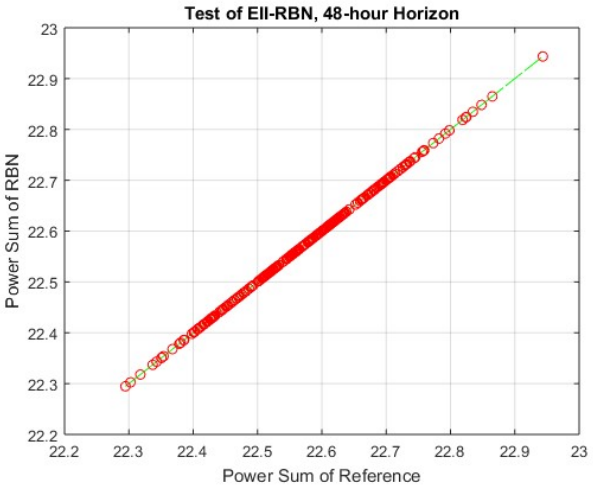

Figure 6. Testing for 48-hour horizon, RBN vs. reference.

D. *Computational Efficiency*

The performance of trained ANN needs to be further tested and compared with some benchmark methods. Conventional approaches of solving network optimization problems are represented by those iterative procedures. They are available in most standard numerical software packages. Since the datasets used in the present work were generated by a well-known procedure Fmincon in Matlab, it is appropriate to compare the trained ANN with Fmincon. The concerned process is referred to as *external testing*, as it uses a different dataset that was unseen in the training phase. According to ANN principles, however, this new dataset must follow the same probability distribution as mentioned in Section II. The simulations were conducted in a small computer equipped with the 3.50 GHz quad-core CPU and 16 GB RAM. This makes the module very feasible for MEL platforms.

The computation time is listed in Table VIII., where the running time is averaged over all samples and the unit is millisecond. It is shown that, by using RBN, the computational efficiency has been significantly increased.

TABLE VIII. PERFORMANCE COMPARISON

	1-hr	48-hr
Fmincon	182.4	889.2
RBN Test 1	0.2351	0.2694
RBN Test 2	0.2203	0.2443
RBN Test 3	0.0482	0.0386

VI. CONCLUSION

Edge computing is becoming an indispensable frontline in modern smart grid. Equipping some lightweight ML features in edge computing represents one of the most important innovations for smart grid. In this paper, we make an initial effort to develop a lightweight ML module based on RBN. The selected case study comes from the emission mitigation program. It is of the increasing importance since the beginning of 2021, as the possibility of participating in *Paris Agreement* is high.

REFERENCES

[1] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: motivations, requirements and challenges", *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 5-20, First Quarter 2013.

[2] W. Meng, R. Ma, and H.-H. Chen, "Smart grid neighborhood area networks: a survey", *IEEE Network*, vol. 28, no. 1, pp. 24-32, Jan.-Feb. 2014.

[3] C. Kalalas, L. Thrybom, and J. Alonso-Zarate, "Cellular communications for smart grid neighborhood area networks: a survey", *IEEE Access*, vol. 4, pp. 1469-1493, April 2016.

[4] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: wireless communication meets machine learning", *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19-25, January, 2020.

[5] R. Yokoyama, S. H. Bae, T. Morita, and H. Sasaki, "Multiobjective generation dispatch based on probability security criteria," *IEEE Transactions on Power Systems*, vol. 3, pp. 317–324, Feb. 1988.

[6] E. Denny and M. O'Malley, "Wind generation, power system operation, and emissions reduction", *IEEE Transactions on Power Systems*, vol. 21, no. 1, pp. 341-347, 2006.

[7] X. Liu and W. Xu, "Minimum emission dispatch constrained by stochastic wind power availability and cost", *IEEE Transactions on Power Systems*, vol. 25, no. 3, pp. 1705-1713, August 2010.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016.

[9] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[10] M. A. Abido, "Environmental/economic power dispatch using multiobjective evolutionary algorithms", *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1529-1537, 2003.

[11] D. B. Das and C. Patvardhan, "New multi-objective stochastic search technique for economic load dispatch," *Proc. IEE Gen. Transm. Dist.*, vol. 145, no. 6, pp. 747-752, 1998.

[12] S. Bofinger, A. Luig, and H. G. Beyer, "Qualification of wind power forecasts," in *Proc. 2002 Global Wind Power Conf.*

[13] A. Fabbri, T. G. S. Roman, J. R. Abbad, and V. H. M. Quezada, "Assessment of the cost associated with wind generation prediction errors in a liberalized electricity market", *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1440-1446, 2005.

[14] H. Bludszuweit, J. A. Dominguez-Navarro, and A. Llombart, "Statistical analysis of wind power forecast error", *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 983-991, 2008

[15] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering* (3rd ed.), Pearson, Upper Saddle River, NJ, 2008.

[16] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products* (7th ed.), Academic Press, 2007.

[17] R. Fletcher, *Practical Methods of Optimization* (2nd ed.), John Wiley & Sons, NY, 1987.

[18] *Neural Network Toolbox*, Matlab, R2018a.

[19] M. J. D. Powell, "Radial basis functions for multivariable interpolation: a review". In J. C. Mason and M. G. Cox (Eds.), *Algorithms for Approximation*, pp. 143–167. Oxford University Press, 1987.