Electric Load Forecasting Model Using a Multicolumn Deep Neural Networks

Ammar O. Hoori, Student Member, IEEE, Ahmad Al Kazzaz, Rameez Khimani, Yuichi Motai, Senior Member, IEEE, and Alex J. Aved

Abstract—In this article, a new approach to short-term load forecasting is proposed using a multicolumn radial basis function neural network (MCRN). The advantage of this new approach over similar models in speed and accuracy is also discussed, especially in regards to renewable generation forecasting. Because weather and seasonal effects have a direct impact not only on load demand but also on renewable energy production, it follows that as the penetration rate of renewable DG increases, the grid will become even more sensitive to weather impacts in the long term. In our approach, we use a k-d tree algorithm to split our feature-rich dataset into dense specialized subsets. These subsets are then trained in parallel as multiple artificial neural networks using a modified error correction algorithm to form the MCRN. This approach reduces the number of hidden neurons, increases the speed of convergence, and improves generalization over similar alternative forecasting methods.

Index Terms—Deep learning, electric forecasting, industrial, neural network, renewable energy.

 NOMENCLATURE

The following notations will be used throughout this article and are provided below for quick reference. Any other symbols will be defined as needed throughout the text.

ANN Artificial neural network.
DG Distributed generation.
ELM Extreme learning machines.
ErrCor Error correction algorithm.
ISO Improved second order.
KNN k-nearest neighbors algorithm.
k-d tree k-dimensional tree.
MCRN Multicolumn RBF network.

I. INTRODUCTION

RENEWABLE distributed generation (DG) has a growing impact on the world’s energy landscape as prices drop and adoption increases. The proliferation of renewable DG, however, comes at the cost of deteriorating short-term load forecasting (STLF) models, which have not yet been adapted for medium- and large-scale renewable DG deployment. STLF has a vital role in managing the operation scheduling of power systems such as economic dispatch, unit commitment, and energy transactions [1]–[4], and therefore is a powerful tool for electricity demand forecasting and achieving cost goals.

In order for the utility to meet energy demand at the lowest possible cost, they use various methods of economic dispatch to optimize their energy economy. Normally, this involves the short-term manipulation of all available energy resources to meet energy demand at the most cost-effective operating points, mainly taking into consideration the cost of operation and availability of that resource. Since the overall system has a combined capacity much larger than what the grid would require, each energy resource allocation can be optimized to provide energy at the approximately lowest cost and risk available, thus reducing the end cost to consumers. Increasing their distributed renewable generation portfolios would boost their system capacity and reduce dependence on volatile fossil fuel costs. This is important to the utility, who has made it a priority to reduce risk and

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPE</td>
<td>Mean absolute percentage error.</td>
</tr>
<tr>
<td>MTLF</td>
<td>Mid-term load forecast: 1mo. &gt; t &lt; 1yr.</td>
</tr>
<tr>
<td>RBFN</td>
<td>Radial basis function network.</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root-mean-squared error.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machines.</td>
</tr>
<tr>
<td>STLF</td>
<td>Short-term load forecast: t &lt; 1mo.</td>
</tr>
<tr>
<td>k</td>
<td>Positive integer typically small.</td>
</tr>
<tr>
<td>( \tilde{x} )</td>
<td>Input Layer ( I ) units, denoted as ( \tilde{x} = [x_1, x_2, \ldots, x_i, \ldots, x_j] ).</td>
</tr>
<tr>
<td>( \theta(\tilde{x}) )</td>
<td>Hidden layer RBF units ( \theta(\tilde{x}) = [\theta_1, \theta_2, \ldots, \theta_h, \ldots, \theta_{11}] ).</td>
</tr>
<tr>
<td>y</td>
<td>Output.</td>
</tr>
<tr>
<td>( w_h )</td>
<td>Weight between ( \theta_h ) and ( y ).</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>Bias, which is the weight between ( \theta_0 = 1 ) and ( y ).</td>
</tr>
<tr>
<td>( c_h )</td>
<td>Center vector of hidden unit ( h ).</td>
</tr>
<tr>
<td>( \sigma_h )</td>
<td>Width of hidden unit ( h ).</td>
</tr>
<tr>
<td>( y_d )</td>
<td>Desired output.</td>
</tr>
<tr>
<td>P</td>
<td>Number of training pairs.</td>
</tr>
</tbody>
</table>
operation expenses through increased scrutiny on planning day-
to-day operations with weather events and more precise load
forecasting strategies.

Power load forecasting is the key process in planning effi-
cient power systems that predict the power load in ahead and
produce fewer losses in terms of energy and costs. Power load
forecasting can be classified into three times frame categories:
STLF, medium-term load forecasting (MTLF), and long-term
load forecasting (LTLF). Their respective forecasts span from
hours to days, weeks to months, or one to several years [5]–[7].
Although LTLF and MTLF can be useful for long-term planning
operations and maintenance, STLF arguably has the most im-
portant role due to its importance in unit commitment. Because
of this, STLF must be fast and accurate to avoid the problems
and costs associated with under or overcommitting resources,
and if properly designed, will be able to mitigate some of the
effects of renewable DG on the grid [8].

The approach used to develop load forecasting techniques
can vary slightly but typically take either a statistical approach
or artificial intelligence (AI) approach [2], [8], [9]. Due to the
nonlinear nature of power consumption, statistical forecasting
models tend to result in lower accuracy [6], whereas AI fore-
casting models remain very popular due to the ability to adapt
nonlinear patterns with various machine learning techniques.

Artificial neural networks (ANNs) have been very popular
in machine learning and pattern detection for a very long time
[10]–[14], and continue to have leaped in improvement. Relative
to statistical methods, ANNs are initially much more complex
to set up, but when implemented for well-suited problems can
realize significant reductions in error. This is primarily because
statistical methods are unable to handle nonlinearity well, and
AI methods are designed to “learn” through problems. For this
reason, AI forecasting methods have now found their way into
use for short-term electric load forecasting (STLF) [6], [9],
[15]–[19]. More specifically, the nonlinearity in power con-
cumption often results from seasonal effects, weather condi-
tions, and varying socioeconomic conditions [18], and makes
accurate statistical forecasting unrealistic and machine learning
methods a stronger choice for fitting unseen or ill-understood
patterns.

ANN has long been used to approach nonlinear learning prob-
lems with great success, but when applied to feature-rich load
forecasting problems, they tend to result in computationally
heavy hidden unit layers. In other words, increasing the com-
plexity of a neural network generates tremendous time delays for
training. Due to this, many studies in this area have had to trim
the number of input features to allow training times to become
manageable, but do so at the cost of compromising the gener-
alization ability of their ANNs [20]. Furthermore, because the
ability of ANNs to fit unseen patterns observed within training
data, or learn, is the primary purpose for its creation and contin-
ued use, it is apparent that an AI forecasting model should in-
clude as many features as possible to increase performance [21].

In addition, due to the influx of renewable DG, weather pat-
terns have become double important in regard to load forecasting
and generation planning. Now, they would not only affect en-
ergy consumption due to seasonal temperature effects but also
the production of renewable DG, thus in effect taking a larger
role in generation planning. For example, Din and Marnerides
[6] used deep neural networks (DNN) for SLTF, and even cites
the effects of “heterogeneous sources” related to weather, while
ultimately choosing not to include it. However, since solar ra-
diation, wind speed, and other weather factors directly affect
both the outputs of renewables and load demand, we feel that
it is necessary to develop a forecasting model that handles a
greater level of complexity to provide desirable levels of gener-
alization at a low time cost [22]. For example, Zhang et al. [23]
introduced a weather condition forecasting model of the photo-
voltaic (PV) power using adaptive regression spline. However,
Ceci et al. [24] used spatial information from the power load
for PV dataset to train an online neural network adaptive model.
Therefore, our goal is to develop a combined approach that
results in a load forecasting model which can provide better in-
sight into the rapid variations of renewable DG sources, and do
so with relatively fast training time and good generalization of
the output.

The rest of the article is organized as follows. Section II pro-
vides an overview of alternative forecasting methods, focusing
on parametric and machine learning methods. Section III intro-
duces the multicolumn radial basis function network (MCRN)
and discusses how it will be applied to achieve both a short-term
load forecast and an accurate estimation of hourly wind and so-
lar energy contributions. Section IV presents the details of the
experiment, including a detailed presentation, comparison, and
discussion of the results. Section V concludes this article.

II. STLF WITH A RADIAL BASIS FUNCTION
NETWORK (RBFN)

The RBFN machine learning model provides a good basis
for STLF, and researchers have sought to improve its speed
and reliability. Recent research developments have enhanced
the performance of this model with faster and more accu-
rate error correction algorithms, deep techniques, and outlier
pruning.

A. RBFN With ErrCor

Due to the complexity of electric load forecasting, compre-
hensive ANN implementations would inherently possess a large
number of hidden neurons, increasing the size of the network and
slowing down the training process. Thus, in order to minimize
or at least reduce training time, careful consideration of ANN
architectures and training algorithms must be considered so that
the ANN remains compact, but also generalizes satisfactorily.
For this reason, an RBFN was chosen for its compact architec-
ture, along with the availability of an existing error correction
algorithm called ErrCor, which was designed to minimize the
number of hidden neurons through a selection of RBF centers
via the most violating vector. In order to accomplish this, one
must first calculate the hidden unit of the RBFN (1), determine
which vector is the maximum violator, and set it as the center for
the next hidden unit. Within each iteration of a new hidden unit
calculation, a system of hidden unit weights \(w_h\), is optimized to a
set error threshold, after which both the hidden layer and hidden
weights can be used to validate the training data or forecast the output of the RBFN (2), given new data.

$$\theta_h(x) = \exp\left(-\frac{||x - c_h||^2}{\sigma_h}\right)$$ (1)

$$y = \sum_{h=1}^{H} w_h \theta_h(x) + w_0$$ (2)

It can be seen that each hidden unit must be calculated exponentially, and therefore, as the number of hidden neurons increases, the number of computations increases, and thus training time increases greatly. Based on this, the ErrCor algorithm was developed to guarantee a smaller number of hidden neurons, and therefore resulting in faster training times. The process of selecting new RBF, hidden neuron centers via the maximum violating vector results in a much more compact neural network, which then generalizes better, and converges much quicker. The inclusion of ErrCor into RBFN load forecasting has shown significant improvement over traditional forecasting methods such as support vector machines (SVM), extreme learning machines (ELM), and improved second-order (ISO) algorithms [17]. Furthermore, Cecati et al. [17] additionally improved the performance of this method through data pruning and limited application of deep learning techniques. Hoori and Motai [10] and Cecati et al. [17] also noted that DNN and deep learning techniques perform desirable when datasets are dense, and generalize better overall, but some DNNs have delays that tend to compound as the number of patterns increase, which is likely to occur for complex problems such as electric load forecasting. Instead, they mention that swapping DNNs with RBFNs, while still applying deep learning techniques, have had success in generating a much more shallow architecture with better generalization properties.

**B. Deep Learning Techniques**

It is also known that training an RBFN has constraints that can affect performance [17], and therefore, the proper training of input features is crucial to a well-functioning neural network. For example, overtraining can lead to poor results from overgeneralization, but undertraining can lead to misrepresentation or under-representation of input features. For this reason, it is very important to consider all the factors that may go into electric load consumption, but also cleverly choose the limits on the number of epochs the algorithm iterates through, effectively limiting the size of the hidden network. However, because ANN training is most successful with high-density datasets, a comprehensive training dataset will guarantee a large hidden network, effectively limiting the speed of training. This is in spite of the use of ErrCor to minimize the hidden layer and results from the excessive number of computations required to generalize such a large set of data.

Hoori and Motai [10], showed that the ErrCor algorithm does indeed guarantee a fast start and acceptable generalization, but as the number of hidden neurons increases incrementally, the delays increase exponentially. In other words, even though the ErrCor algorithm succeeds in minimizing the hidden layer, at a certain threshold of input feature density, the speed of convergence of the training will typically approach a bottleneck. This finding suggests that for the RBFN, smaller datasets train faster, and higher density datasets generalize better.

Hoori and Motai further expounded on this idea by suggesting that delays caused by feature-dense datasets can be effectively eliminated by creating smaller high-density subsets of data from the original set, and use them to train multiple, specialized RBFNs in parallel. In this way, the total number of epochs required to generate the hidden layer remains relatively small, and each resulting neural network will be generalized well such that it would effectively represent a specific learned pattern. This novel method is known as the MCRN and is represented in the block diagram of Fig. 1.

During training, the dataset is divided into smaller subset using k-d tree algorithm. Each subset is used to train an individual ANN (RBF network in this case). Those Individual ANNs will be responsible for any data that belongs to the designated subsets. Fig. 1 represents the MCRN detailed structure during testing. With a new test data vector $\tilde{x}$, the input subset selector uses K-nearest neighbors (KNN) method to select only K neighbors’ points. These points were used in the training of some of the NN(s). Only the neighbors NNs will be fired and each will give a suggested output $\{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_N\}$ to the NNs output combiners. The output combiner will average the sum of these designated K outputs to produce the MCRN output $\tilde{y}$.

The MCRN serves as the foundation for the forecasting model described in the rest of this article and has been prescribed to this particularly complex issue of STLF with renewable DG due to its ability to handle large amounts of features, in relatively short training time. Dividing the STLF dataset into a subset using K-d tree and train then in parallel will increase the speed of training. Using ErrCor as the training algorithm for each individual ANN will give the process improved specialties over those subsets. STLF dataset needs fast decision maker during testing, which MCRN suggests improvement in both training and testing. The theory of operation for the MCRN will be discussed, which will be followed by a benchmark comparison of results against other STLF models.
III. MCRN FORECASTING MODEL

Fig. 1 depicts the three main layers of the MCRN internal structure. The Individual NNs are the trained RBFNs stacked in parallel, the input subset selector that selects the appropriate NNs to be fired based on the new input entry, and the NNs output combiner that average sum only the outputs of the fired NNs [10]. The parallel RBFNs are pretrained by splitting a much larger training dataset into multiple smaller training sets using the k-d tree chopping algorithm. These subsets are used individually to train specialized ANNs using the modified ErrCor algorithm which is used to make 24 h load forecasts. This process of training and testing will be explained shortly in the following sections.

A. k-d Tree Chopping

When the training dataset is prohibitively large, time consuming and wasteful calculations can occur during the training of the RBFN hidden units. Although the ErrCor algorithm already minimizes the number of hidden units to reduce delays, the complexity of electric load forecasting forces us to modify this approach. By dividing our training dataset into more manageable specialized subsets, we can turn a large structured, slow-training ANN into parallel faster training ANNs, as shown in Fig. 1.

Using the k-d tree algorithm specified, as shown in Algorithm 1, we chop our multidimensional dataset \( \epsilon \) times using our selected six input features. The subsets are then trained into \( N = 2^\epsilon \) specialized ANNs. Considering the density as the number of positive-value labels over the whole numbers of dataset’s labels, the k-d tree algorithm chops the dataset into half, using the median of a certain feature that keeps the density of the subsets approximate to the original dataset density.

This process is repeated for each new subset for \( \epsilon \) times, as shown in Algorithm 1. The k-d tree algorithm also prevents zero-data subsets and ensures that the distribution of data allows for similar densities across subsets. This is important because sparse training sets typically result in poorly trained ANNs, and therefore poor generalization. Therefore, creating and training multidimensional subsets with similar densities should provide better generalization for training purposes.

Training of the system must be completed offline and is done using a training set \( \{x_p, y_p\} \), meaning that multiple training sets should be provided to train the network robustly. We can first calculate the error for each input pattern, and then use it to calculate the networks root-mean-square error (RMSE), which will serve as the benchmark comparison to other load forecasting methods.

It is also important to mention that the original data subset was normalized before chopping. The input features were normalized to \([-1, 1]\) and then the output vector was normalized to \([0, 1]\). Once the subsets have been chopped, they are trained as multiple RBFNs with the ErrCor algorithm and stacked in a multicolumn structure, the MCRN.

ErrCor [12] algorithm train the ANN structure by inserting the maximum violator vector from the training dataset to be the new hidden unit center vector at each epoch. The maximum violator vector is the input vector that produced the maximum error of the ANN compared with the desired output. This vector should be excluded from the input dataset and the training will continue until the maximum error reached the desired tolerance.

B. MCRN Load Forecasting

During data validation and forecasting, only the applicable \( k \) ANNs will be selected based on the KNN algorithm, which are then combined to provide a forecast. The Euclidean distance between the test vector and all training datasets will be calculated, and the \( k \) smallest Euclidean distances will be used to select the applicable \( k \) ANNs. As shown in Fig. 2, each \( k \) point corresponds to a subset, which was used in training a specialized RBFN. These distances \( d_k \) can be calculated using the following equation:

\[
d_k = \min_{1..k} \left( \sum_{i=1}^{T} (x_i - x^p_i)^2 \right)^{1/2}
\]

This algorithm is used to make 24 h load forecasts. The Euclidean distance between the test vector and all training datasets will be calculated, and the \( k \) smallest Euclidean distances will be used to select the applicable \( k \) ANNs. As shown in Fig. 2, each \( k \) point corresponds to a subset, which was used in training a specialized RBFN. These distances \( d_k \) can be calculated using the following equation:

\[
d_k = \min_{1..k} \left( \sum_{i=1}^{T} (x_i - x^p_i)^2 \right)^{1/2}
\]

Algorithm 1: k-d Chopping Process.

Inputs: \( \epsilon, \% \) the number of chopping processes \( \{x, y\}, \% \) the number of Individual NNs

Output: \( N = 2^\epsilon, \% \) number of Individual NNs

Initialization
1: Compute original dataset density \( D \)
2: for \( e = 1 \) to \( \epsilon \) do, \% \( e \) = number of chops
3: for \( s = 1 \) to \( (2^e - 1) \) do, \% \( s \) = number of subsets
4: Find each feature median \( x_i \)
5: Computer average density \( D_i \) for
6: Select feature \( i \) with minimum \( abs(Di - D) \)
7: Chop dataset along median \( x_i \) into two subsets
8: end for
9: end for
10: for \( s = 1 \) to \( N \) do
11: Train the \( s \) Individual NN using \( s \) subset
12: end for
After the ANNs are selected, the input is tested in those specific ANNs using (4), resulting in $k$ outputs. The resultant outputs from the contributing ANNs are combined to calculate the overall output $\tilde{y}_k$ by averaging the vector of outputs $\tilde{y}_k$ as in (5). The overall process on MCRN is listed in Algorithm 2.

$$y_k = \sum_{h=0}^{H} w_h^k \theta_h^k (x)$$ \hfill (4)

$$y = \frac{1}{k} \sum_{k} \tilde{y}_k$$ \hfill (5)

Algorithm 2 explains the run-time structure shown in Fig. 1, where the input subset selector uses the Euclidean distance to determine the $k$ nearest neighbors using the KNN method. These neighbors belong to subsets, which are previously used to train $k$ individual NNs. These neighbors NNs are considered the specialist pretrained networks that can suggest the desired output for that specific test input vector. Therefore, their suggestion is used to contribute toward the overall output.

### C. Renewable DG Forecasting

The intermittent nature of energy sources such as wind and solar can be problematic when producing STLFs, but by developing a method of accurate renewable DG forecasting, we can increase grid resiliency [25]. Energy planning can be greatly impacted through increased precision in load estimation, and the preprocessing of historical load data has been shown to positively affect it [2]. By training the MCRN algorithm with historical data, the MCRN can identify patterns through deep learning techniques and modify the weights of certain features accordingly. Thus, we can evaluate the impact of specific features independently and collectively to determine their contribution to the STLF, which will provide us with a trained system that results in the lowest possible RMSE, given by (10).

The solar radiation can be estimated precisely through the PVGIS [26]. The methods used to calculate the solar radiation from satellite images to estimate the influence of clouds on solar radiation. Clear sky conditions (i.e., no clouds) using the theory of radiative transfer in the atmosphere together with data on how much aerosols (dust, particles, etc.) there are in the atmosphere and the concentration of water vapor and ozone, both of which tend to absorb radiation at particular wavelengths. The total radiation is then calculated from the cloud albedo and the clear-sky irradiance. This may cause a huge computational complexity.

As an alternatively easy method, for the 24 h STLF, a specific substation power output data will be trained against applicable input features. For this study, the features in Table I are considered. However, in order to understand the impacts renewable DG may have on the grid, we also forecast 24 h solar power and wind power for which input features are listed in Tables II and III, respectively:

$$\text{Im} = \text{I}_i \left[ \cos(\alpha) \sin(\beta) \cos(\psi - \theta) + \sin(\alpha) \cos(\beta) \right].$$ \hfill (6)

Solar radiation, as shown in Fig. 3, can typically be modeled by (6), which takes into consideration the solar insolation $I_i$, or the amount of sunlight incident to the surface of the solar cell, sun elevation angle $\alpha$, module tilt angle $\beta$, $\theta$ sun azimuth angle, and module azimuth angle $\psi$. All of these factors take
into consideration the month, date, year, hour, minute, second, geographical location, and placement angles. This means that the solar insolation for a particular time on a specific day and geographical location can be generally approximated with great ease, and thus, one would assume that solar DG could be easily forecasted.

Unfortunately, this is not the case due to other factors such as cloud coverage, which can cause intermittent drops in production since solar power is directly dependent on insolation levels; therefore, the expected solar insolation can only be considered a constraining factor in determining the solar power forecast. This constraint makes it well suited to use as an input feature while training an ANN that can produce solar power forecasts, along with weather factors that can also affect the production of solar power.

Similarly, wind power is dependent on the available wind resource, which is currently assessed by case studies on a particular geographical location. Case studies performed for collected wind data can be used to create wind speed distributions such as the Rayleigh wind speed distribution, as seen in Fig. 4, which is described by (6), where $b$ is the shape parameter and $V$ is the wind velocity.

$$ PW = C_p \frac{1}{2} \rho AV^3. \quad (8) $$

The equation described in (8) is used to determine the electric power that can be extracted from the wind. For (8), $C_p$ is the maximum power coefficient, $\rho$ is the air density, and $A$ is the cross-sectional area swept by the rotor. $C_p$ is a value between 0.25 and 0.593, which is obtained from applying Betz’s theoretical limit of extraction of power from a fluid, which states that only about two-thirds of the total applied energy of a fluid can be converted to power.

When used in conjunction with each other, (7) and (8) can be used to determine a guideline for wind power generation at a site annually but cannot provide much else useful for a short-term wind power prediction. Therefore, we again would benefit from a trained ANN that takes these constraints into consideration but does not strictly rely on them for next-day generation planning.

For this article, we will forecast and utilize both 24-h solar and wind data to accurately predict the available solar and wind resource for renewable DG production in addition to load demand forecasts. By doing so, we hope to combine all three forecasts to form a composite forecast of load demand and less the predicted renewable generation from solar and wind. The results will vary based on the given installation capacity for these renewable DG sources.

**IV. EXPERIMENTAL RESULTS**

The following sections detailing the experimental results are characterized as follows. Section IV-A provides a brief overview of the input features and expected the output of each neural network case study, including the methods used to compare the results. Section IV-B provides a comprehensive overview of our results when compared against other RBFN forecasting models, including the standard ErrCor forecasting model. This is then followed by a detailed analysis of the MCRN forecasting model in Section IV-C, and Section IV-D with our Renewable DG forecasting model which includes the adapted MCRN model.

**A. Data and Criteria**

For experimental verification, the New England ISO hourly load profile data [27] from 2003 to 2015 were utilized to train the MCRN and test against the modified and unmodified ErrCor RBFN forecasting models, and then the model was further modified to generate a case study on high penetration renewables. Weather data acquired from [28] for years ranging from 1991 to 2015 were used to produce a renewable generation forecast for solar and wind power. Dataset is divided into 80% training and 20% testing.

The input features selected for the training set were also derived from the New England ISO data [18], and selected
to be the same features used in [17] and [18] to provide a
direct basis for comparison for our results. Input features for
the renewable DG forecasting, however, were determined by
performing multiple case studies to determine the relevant input
features suitable for solar and wind generation forecasting.

Table I provides the input features for the 24 h load forecast.
These features include a description of the time of the day, a
holiday or not, and the dry bulb and dew point temperatures
and a previous value of 24 h (1 day) and 156 h (approx. 1 week)
power load, whereas Tables II and III provide the parameters for
the solar generation and wind power generation, respectively.
These measurements are normalized between 1 and −1.

In direct comparison to existing methods, our results will be
compared using mean-square error (MSE or ECR) and RMSE
methods

\[
MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{f_i - y_i}{f_i} \right|.
\]  

(9)

The mean absolute percent error (MAPE) is described by (9),
where \( f_i \) is the observed output and \( y_i \) is the forecast output.
The result of this equation is that the average of the absolute
value of errors is taken

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f_i - y_i)^2}.
\]  

(10)

Alternatively, using (10), we can compare the standard devi-
ation of observed and forecasted values, again where \( f_i \) is the
observed output and \( y_i \) is the forecast output, but is instead used
to calculate the standard deviation. By calculating the MAPE
and RMSE of all three methods of load forecasting, we can
compare the results of our forecasting method as a benchmark
against the existing methods of STLF.

B. RBFN With ErrCor

The ErrCor and modified ErrCor algorithms were indepen-
dently verified using Matlab code provided in [17]. The ErrCor
and modified ErrCor methods of training the RBFN used in [17]
were randomly split into training and validation subsets, trained,
and then validated. The primary indicators of performance used
were the RMSE, and the time per neuron calculation as seen in
Figs. 5 and 6.

Fig. 5 depicts a one week forecast for the normalized power
of the regular ErrCor and the modified algorithm presented in
[17]. Because the RMSE was significantly for our results, we proceeded
to compare the speed of the MCRN versus the modified ErrCor
in Fig. 6. The results in Fig. 6 show that as we increase the num-
ber of divisions \( N \) to create \( 2N \) subsets, our training time for
our MCRN significantly decreases. For comparison, the lowest
RMSE was achieved for \( N = 3 \) which results in eight special-
ized ANNs. The next two bests were achieved for \( N = 2 \) and
\( N = 1 \), which represent four and two subsets, respectively.
The worst speed and RMSE were given by the ErrCor algorithm,
which was trained without subset splitting. Overall, the results
indicate that as the number of hidden unit neurons increases,
the time required for convergence increase, and also that as the
number of subsets, and therefore parallel ANNs, increase, the
time for convergence decreases.

C. MCRN Forecasting

Fig. 7(a) shows a combined loads forecast which is the result
of the KNN subset selector, which searches for the optimal
parallel ANN based on its three nearest neighbors. Fig. 7(b)
shows the individual outputs of the individual ANNs before
the subset selection process takes place. As we can see from the
column, some of the ANNs have the best forecast at different
times, and thus become more optimal for subset selection.

Table IV provides the MAPE for the best iteration for 50
neurons, which were processed individually for each day of the
week. By doing this, we can see that it is unnecessary to perform
this step again as performed by Cecati et al. [17] because the
MCRN already reliably performs this operation innately when
the correct subset division is performed. Table IV also shows the
errors relative to specific days of the week over the years 2010
and 2011. This table shows that we can achieve a better forecast
when similar days of the week are examined, as opposed to the
entire dataset at once.

Table V provides a benchmark comparison against the best
RMSE achieved for each method of forecasting. Only the best
resulting MAPE is shown for SVM, ELM, ISO, ErrCor, Modified
ErrCor, and our MCRN method.
The results in Table V indicate that the MCRN is not as accurate as of the previous methods; however, it is worth noting that the MCRN model took a total of 10 min to train as opposed to hours for the other two methods. Although accuracy is important in load forecasting, still speed is a very important factor, especially when the power generating a decision is in highly critical risk such as nuclear or hydraulic plants. The MCRN’s input features would have to be modified to include more input features in order to achieve a lower error.

D. Renewable DG Forecasting

Fig. 8 shows a “7 × 3” forecast, which uses seven inputs to derive a total 24 h forecast, a solar DG forecast, and a wind DG forecast. The solar and wind generation forecasts are derived from the respective renewable resource forecasts multiplied by the overall capacity of the renewable DG systems. For the U.S., renewable energy accounts for 9.9% of the total generation capacity, for which solar accounts for 0.45% and wind accounts for 1.71%. Because these renewables are the fastest growing in use, but also the most intermittent of all energy sources, it is worth taking a look at the composition of solar and wind against the remaining load forecast.

The wind resource observed in Figs. 8 and 9 incorporate the Rayleigh wind speed distribution for the same geographical location as described by (7) and (8). The Betz limit efficiency of 0.593 was used to determine the power from wind conversion,
but it must be noted that realistically each substation service area would have some unique, nominal efficiency factor typically between 0.25 and 0.45.

Table VI provides the best forecast’s level of error and the absolute error from the actual observed energy production. As we can see, the magnitude of the error is very small and allows for a reasonable prediction for the share of other generation sources. By doing this, we can optimize the economic dispatch of DG sources with a reliable inclusion of renewable DG sources, thus lowering costs and increasing reliability.

V. CONCLUSION

As the proliferation of renewable DG continues, fast and accurate STLF grows in importance. Existing forecasting models for STLF are computationally taxing and cannot support the density in features that might be required to study the granular details of a renewable DG system. By splitting an input-feature heavy dataset into multiple datasets, and training them in parallel as individual RBFNs, in this article, we showed that we can greatly reduce computation time while maintaining reasonable levels of generalization found in comparable RBFN forecasting models.

The added benefit of training and testing renewable DG forecasts can be done rapidly to reevaluate STLFs quickly when the need arises. These renewable DG forecasts provide insight into potential energy shortages or surpluses and can eventually be reliably incorporated into energy planning and economic dispatch calculations. Currently, the types of data for renewable DG forecasting need to be selectively pruned to determine the best features required for forecasting, such as the approach of using maximum, minimum, and average values of wind speed and temperature in [29].

REFERENCES


### Table VI

<table>
<thead>
<tr>
<th>Day</th>
<th>Power Gen.</th>
<th>RMSE</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wind Resource</td>
<td>240 kW</td>
<td>39.3</td>
<td>90</td>
</tr>
<tr>
<td>Solar Resource</td>
<td>100 kW</td>
<td>40</td>
<td>90</td>
</tr>
<tr>
<td>Other Sources</td>
<td>700 kW</td>
<td>10.1</td>
<td>90</td>
</tr>
<tr>
<td>Total</td>
<td>1640 kW</td>
<td>270</td>
<td></td>
</tr>
</tbody>
</table>
Ammar O. Hoori (S’17) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Baghdad, Baghdad, Iraq, in 1999 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2019.

From 2008 to 2013, he was a Teacher and a Researcher with Computer Engineering Department, University of Baghdad. From 2014 to 2019, he was a Research Assistant with Sensory Intelligence Lab, Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond. His current research interests include machine learning, neural networks, computer networks, and distributed systems.

Ahmad Al Kazzaz received the M.S. degree in electrical and computer engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2014.

He is currently a Software Engineer for Nokia, Espoo, Finland.

Rameez Khimani received the M.S. degree in electrical and computer engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2017.

He is currently a Systems Engineer with Northrop Grumman Corporation, Linthicum, MD, USA.

Yuichi Motai (S’00–M’03–SM’12) received the B.Eng. degree in instrumentation engineering from Keio University, Tokyo, Japan, in 1991, the M.Eng. degree in applied systems science from Kyoto University, Kyoto, Japan, in 1993, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2002.

He is currently an Associate Professor of Electrical and Computer Engineering with Virginia Commonwealth University, Richmond, VA, USA. His research interests include the broad area of sensory intelligence, particularly in data analytics, pattern recognition, computer vision, and sensory-based robotics.

Alex J. Aved received the B.A. degree in computer science and mathematics from Anderson University, Anderson, IN, USA, in 1999, the M.S. degree in computer science from Ball State University, Muncie, IN, USA, in 2001, and the Ph.D. degree in computer science, especially real-time multimedia databases, from the University of Central Florida, Orlando, FL, USA, in 2013.

He is currently a Technical Advisor with the Air Force Research Laboratory Information Directorate, Rome, NY, USA. His research interests include multimedia databases, stream processing (via CPU, GPU, or coprocessor) and dynamically executing models with feedback loops incorporating measurement, and error data to improve the accuracy of the model.