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.
DNN	Deep neural networks.
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.
MAPE	Mean absolute percentage error.
MTLF	Mid-term load forecast: $1mo. > t < 1yr$.
RBFN	Radial basis function network.
RMSE	Root-mean-squared error.
SVM	Support vector machines.
STLF	Short-term load forecast: $t < 1mo$.
k	Positive integer typically small.
$\tilde{\mathbf{x}}$	Input Layer I units, denoted as $\tilde{\mathbf{x}} = [x_1, x_2, \ldots, x_i, \ldots, x_I]$.
$\boldsymbol{\theta}(\tilde{\mathbf{x}})$	Hidden layer RBF units $\boldsymbol{\theta}(\tilde{\mathbf{x}}) = [\theta_1, \theta_2, \ldots, \theta_h, \ldots, \theta_H]$.
y	Output.
w_h	Weight between θ_h and y.
w_0	Bias, which is the weight between $\theta_0 = 1$ and y.
$\mathbf{c}_h$	Center vector of hidden unit h.
σ_h	Width of hidden unit h.
y_d	Desired output.
P	Number of training pairs.

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

Manuscript received February 14, 2019; revised May 24, 2019; accepted August 18, 2019. Date of publication September 12, 2019; date of current version March 31, 2020. This work was supported in part by the Higher Committee of Education Development of Iraq and in part by the U.S. Air Force Research Laboratory Summer Faculty Fellowship Program. *(Corresponding author: Yuichi Motai.)*

A. Hoori, A. A. Kazzaz, R. Khimani, and Y. Motai are with the Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: hooriao@vcu.edu; alkazzazaz@vcu.edu; khimanir@vcu.edu; ymotai@vcu.edu).

A. J. Aved is with REID, Air Force Research Laboratory, Rome, NY 13441 USA (e-mail: alexander.aved@us.af.mil).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2019.2939988

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 efficient 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 important 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 forecasting 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 consumption often results from seasonal effects, weather conditions, 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 problems 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 complexity 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 generalization 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 continued use, it is apparent that an AI forecasting model should include as many features as possible to increase performance [21].

In addition, due to the influx of renewable DG, weather patterns have become double important in regard to load forecasting and generation planning. Now, they would not only affect energy 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 radiation, 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 generalization at a low time cost [22]. For example, Zhang *et al.* [23] introduced a weather condition forecasting model of the photovoltaic (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 insight 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 provides an overview of alternative forecasting methods, focusing on parametric and machine learning methods. Section III introduces 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 solar 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 accurate error correction algorithms, deep techniques, and outlier pruning.

A. RBFN With ErrCor

Due to the complexity of electric load forecasting, comprehensive 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 architecture, 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.

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

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

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 desirably 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.

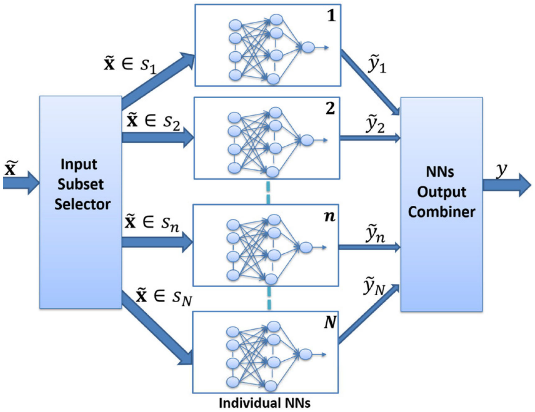


Fig. 1. MCRN topology with *N* specialized ANNs and a single output. Each ANN was trained from an individual subset, specifically chosen to be dense for best generalization.

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 NN* (RBF network in this case). Those Individual NNs 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 $\tilde{\mathbf{x}}$, 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 $\{\tilde{y}_1, \tilde{y}_2, \ldots \tilde{y}_n, \ldots \tilde{y}_N\}$ to the *NNs output combiners*. The output combiner will average the sum of these designated K outputs to produce the MCRN output 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.

Algorithm 1: k-d Chopping Process.

```
Inputs: $\epsilon$, % the number of chopping processes
        $\{\mathbf{x}, y_d\}$,% the training dataset
Output: $N = 2^{\epsilon}$, % number of Individual NNs
    Initialization
Compute original dataset density $D$
for $c = 1$ to $\epsilon$ do, % $c =$ number of chops
    for $s = 1$ to $\epsilon(2^c - 1)$ do, % $s =$ number of subsets
        Find each feature median xi
        Computer average density $D_i$ for
        Select feature i with minimum abs(Di-D)
        Chop dataset along median $x_i$ into two subsets
    end for
end for
for $s = 1$ to $N$ do
    Train the $s$ Individual NN using $s$ subset
end for
```

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 fire 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 ϵ 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 ϵ 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.

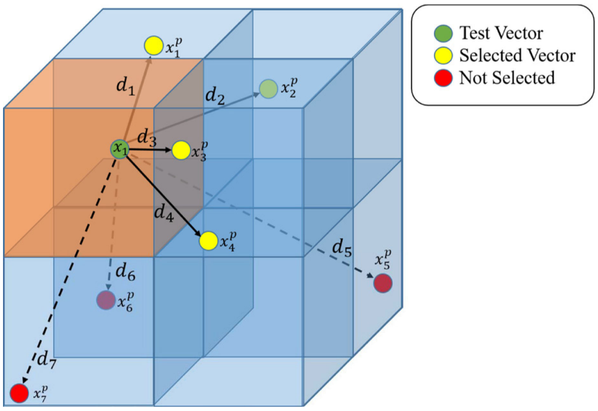


Fig. 2. Example of *N*-chopped k-d tree being used to determine d_k for the output combiner. For each test input vector *x*, the most, relevant ANNs will be used to calculate the output.

Training of the system must be completed offline and is done using a training set $\{\mathbf{x}_p,\ y_p\}$, 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:

$$\begin{aligned} d_k &= \min_{1\ldots k} \left(||\, x - x^p \,||\right) \\ &= \min_{1\ldots k} \left(\sqrt{\sum\nolimits_{i=1}^{I} \left(x_i - x_i^p\right)^2}\right). \end{aligned} \tag{3}$$

Algorithm 2: NN Testing Process.

```
Input:   $\tilde{\mathbf{x}}$, % Testing input.
Output:  $y$, % Testing output.
  Initialization:
Load $2^N$ NNs
for $n = 1$ to $P$ do
  Find the $K\text{set}$ nearest training input, $\mathbf{x}$, to $\tilde{\mathbf{x}}$
  using KNN
end for
for each neighbor, $k \in K\text{set}$ do
    Apply $\tilde{\mathbf{x}}$ to the $k$ NN neighbor and calculate its
    output, $\hat{y}_k$
end for
Compute average for the outputs $\hat{y}_k$ as: $y = \frac{1}{k}\sum_k \hat{y}_k$
```

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 $\tilde{y}$, by averaging the vector of outputs y_k as in (5). The overall process on MCRN is listed in Algorithm 2.

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

$$y = \frac{1}{k} \sum_k y_k. \tag{5}$$

Algorithm 2 explains the run-time structure shown in Fig. 1, where the input subset selector uses the Euclidian 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.

TABLE I
24 H LOAD PARAMETERS [17]

Input	Input Description	Format
1	Hour of Day	(1 - 24)
2	Holiday	(0|1)
3	Dry Bulb Temperature	*(F)*
4	Dew Point Temperature	*(F)*
5	24 Hour Lagged Load	*(kW)*
6	Previous 24 *hr* Average Load	*(kW)*
7	Previous 168 *hr* Lagged Load	*(kW)*
Output	**Output Description**	**Format**
1	24 *hr* Ahead Load	*(kW)*

TABLE II
SOLAR INSOLATION PARAMETER [18]

Input	Input Description	Format
1	Hour of Day	(1 - 24)
2	Day of Year	(0-365)
3	Precipitation	(0|1)
4	Wind Speed	*(m/s)*
Output	**Output Description**	**Format**
1	24 *hr* Solar Insolation	*(kW/m2)*

TABLE III
WIND SPEED PARAMETERS [19]

Input	Input Description	Format
1	Hour of Day	(1 - 24)
2	Day of Year	(0 - 365)
3	Precipitation	(0|1)
4	Wind Speed	*(m/s)*
5	Temperature	*(F)*
6	Air Pressure	*(d)*
Output	**Output Description**	**Format**
1	Wind Speed	*(m/s)*

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:

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

Solar radiation, as shown in Fig. 3, can typically be modeled by (6), which takes into consideration the solar insolation *Ii*, or the amount of sunlight incident to the surface of the solar cell, sun elevation angle α, module tilt angle β, θ sun azimuth angle, and module azimuth angle ψ. All of these factors take

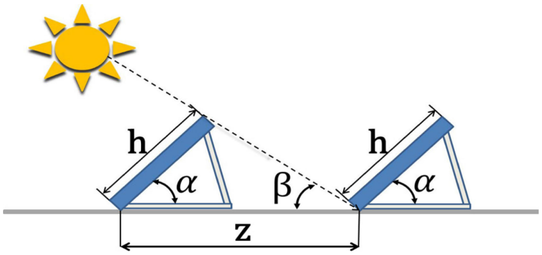


Fig. 3. Example shows how solar resources attained from solar panels are dependent on various types of radiation. Equation (5) models these behaviors into one simple equation.

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

$$p\left(V\right) = \frac{2V}{b^2} e^{\frac{-V^2}{b^2}}. \tag{7}$$

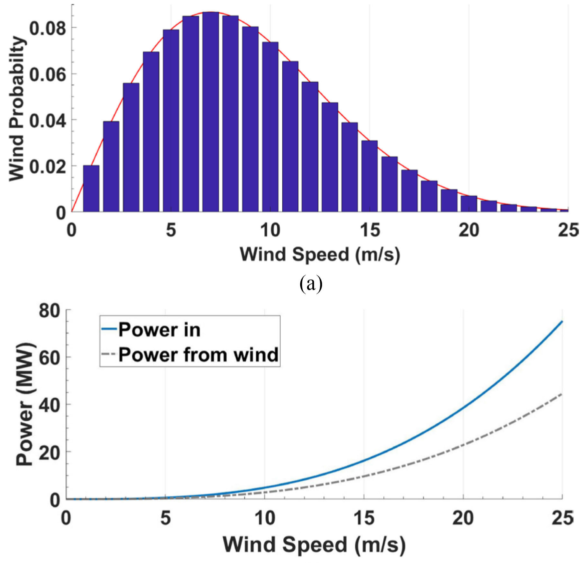
Similarly, wind power is dependent on the available wind resource, which is currently assessed by case studies on a particular geographical location. Case studies performed for collected wind data can be used to create wind speed distributions such as the Rayleigh wind speed distribution, as seen in Fig. 4, which is described by (7), where b is the shape parameter and V is the wind velocity

$$PW = C_p \, \frac{1}{2} p \; AV^3. \tag{8}$$

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, ρ 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, less the predicted renewable generation from solar and wind. The results will vary based on the given installation capacity for these renewable DG sources.



(a)

(b)

Fig. 4. (a) Example of a Rayleigh wind speed distribution which is described by (6), used to show wind statistics for a particular geographic location. (b) Power in the wind diagram is described by (8). Power input to the wind turbine can only result in a theoretical maximum output of 0.593 of the power inputs due to Betz limit.

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 based 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

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{fi - yi}{fi}\right|. \tag{9}$$

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

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(f_i - y_i\right)^2}. \tag{10}$$

Alternatively, using (10), we can compare the standard deviation 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 independently 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 lower, 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 number 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 specialized 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.

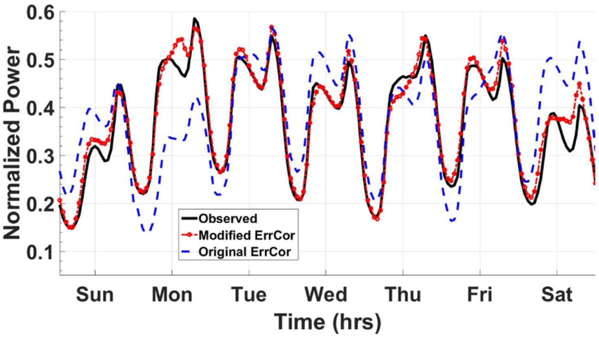


Fig. 5. Seven days electric load forecast using 50 hidden neurons, for the original and modified ErrCor algorithm. Because the modified ErrCor algorithm was significantly better than the original, analysis on the original algorithm was discontinued.

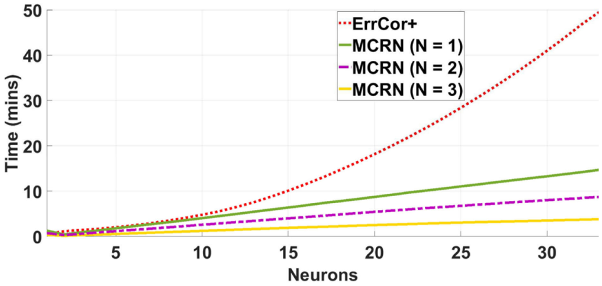


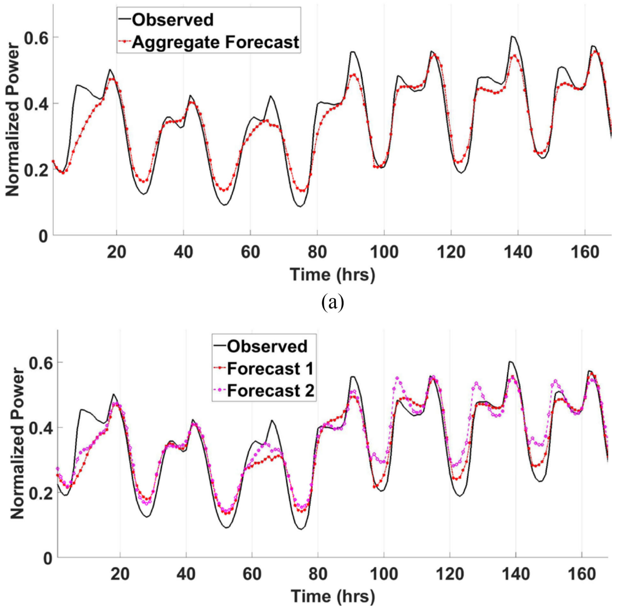
Fig. 6. Training times for the modified ErrCor model versus $N = 1, 2, 3$ subset divisions. As the number of subsets of the original dataset increases, the training time drastically decreases.

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 graph, 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.



(a)

(b)

Fig. 7. (a) Single forecast derived from the MCRN subset selector. (b) MCRN load forecasts for $2N$ subsets. As the subsets get split, the speed of processing increases as well as the RMSE.

TABLE IV
MAPE Over Two Tears With 50 RBFs

Day	Y2010	Y2011
Sunday	2.05	2.47
Monday	2.13	2.56
Tuesday	1.57	1.95
Wednesday	1.68	1.81
Thursday	1.84	1.94
Friday	1.69	1.79
Saturday	1.53	1.65
Overall	**1.78**	**2.03**

TABLE V
Benchmark Performance Against Selected Forecasting

Forecasting Method	Neurons	MAPE
SVM [17]	9040	2.07
ELM [17]	2500	2.19
ISO [17]	70	2.20
ErrCor [17]	150	1.77
Modified ErrCor [17]	150	1.77
MCRN	50	4.59

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.

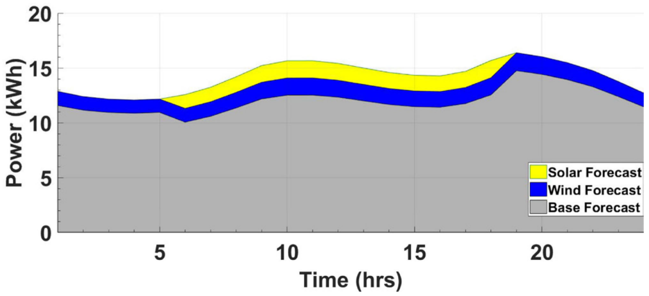
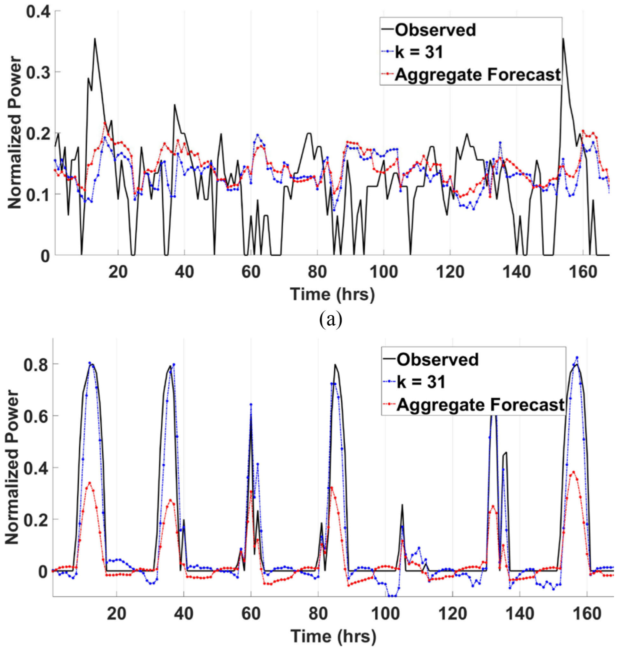


Fig. 8. Predicted renewable DG output against all other sources of energy. This graph is meant to depict the composition of solar and wind energy forecasts against the total load forecast for this particular energy market.



(a)

(b)

Fig. 9. (a) Predicted and observed wind resource forecasts. (b) Predicted and observed solar resources forecasts.

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,

TABLE VI
TOTAL RENEWABLE DG CONTRIBUTION

Day	Power Gen.	RMSE	Time
Wind Resource	240 kW	39.3	90
Solar Resource	100 kW	40	90
Other Sources	700 kW	10.1	90
Total	**1040 kW**	**270**	

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

[1] S. H. Ling, F. H. F. Leung, H. K. Lam, and P. K. S. Tam, "Short-term electric load forecasting based on a neural fuzzy network," *IEEE Trans. Ind. Electron.*, vol. 50, no. 6, pp. 1305–1316, Dec. 2003.

[2] X. Wang, W.-J. Lee, H. Huang, R. L. Szabados, D. Y. Wang, and P. Van Olinda, "Factors that impact the accuracy of clustering-based load forecasting," *IEEE Trans. Ind. Appl.*, vol. 52, no. 5, pp. 3625–3630, Sep./Oct. 2016.

[3] G. Černe, D. Dovžan, and I. Škrjanc, "Short-term load forecasting by separating daily profiles and using a single fuzzy model across the entire domain," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7406–7415, Sep. 2018.

[4] H. Sheng, J. Xiao, Y. Cheng, Q. Ni, and S. Wang, "Short-term solar power forecasting based on weighted Gaussian process regression," *IEEE Trans. Ind. Electron.*, vol. 65, no. 1, pp. 300–308, Jan. 2018.

[5] I. Rahman and S. Moghram, "Analysis and evaluation of five short-term load forecasting techniques," *IEEE Trans. Power Syst.*, vol. 4, no. 4, pp. 1484–1491, Nov. 1989.

[6] G. M. U. Din and A. K. Marnerides, "Short term power load forecasting using deep neural networks," in *Proc. Int. Conf. Comput., Netw. Commun.*, 2017, pp. 594–598.

[7] E. A. Feinberg and D. Genethliou, "Load forecasting," in *Applied Mathematics for Restructured Electric Power Systems*. Boston, MA, USA: Springer, 2005.

[8] D. Singh, R. K. Misra, and D. Singh, "Effect of load models in distributed generation planning," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 2204–2212, Nov. 2007.

[9] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Comparison of machine learning methods for estimating energy consumption in buildings," in *Proc. Int. Conf. Probabilistic Meth. Appl. Power Syst.*, 2014, pp. 1–6.

[10] A. O. Hoori and Y. Motai, "Multicolumn RBF network," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 4, pp. 766–778, Apr. 2018.

[11] W. Kaminski and P. Strumillo, "Kernel orthonormalization in radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1177–1183, Sep. 1997.

[12] H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 10, pp. 1793–1803, Oct. 2014.

[13] D. Cireşan and U. Meier, "Multi-column deep neural networks for offline handwritten Chinese character classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–6.

[14] R. Mall, V. Jumutc, R. Langone, and J. A. K. Suykens, "Representative subsets for big data learning using k-NN graphs," in *Proc. IEEE Int. Conf. Big Data*, 2014, pp. 37–42.

[15] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *Proc. IEEE 26th Int. Symp. Ind. Electron.*, 2017, pp. 1483–1488.

[16] E. Bećirović and M. Ćosović, "Machine learning techniques for short-term load forecasting," in *Proc. 4th Int. Symp. Environ. Friendly Energies Appl.*, 2016, pp. 1–4.

[17] C. Cecati, J. Kolbusz, P. Różycki, P. Siano, and B. M. Wilamowski, "A novel RBF training algorithm for short-term electric load forecasting and comparative studies," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6519–6529, Oct. 2015.

[18] L. Xiao-Fei and S. Li-Qun, "Power system load forecasting by improved principal component analysis and neural network," in *Proc. IEEE Int. Conf. High Voltage Eng. Appl.*, 2016, pp. 1–4.

[19] S. R. Khuntia, J. L. Rueda, and M. A. M. M. van der Meijden, "Forecasting the load of electrical power systems in mid- and long-term horizons: A review," *IET Gener. Transmiss. Distrib.*, vol. 10, no. 16, pp. 3971–3977, 2016.

[20] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—a comparative study," *IEEE Trans. Ind. Inform.*, vol. 8, no. 2, pp. 228–240, May 2012.

[21] M. Rafiei, T. Niknam, and M.-H. Khooban, "Probabilistic forecasting of hourly electricity price by generalization of ELM for usage in improved wavelet neural network," *IEEE Trans. Ind. Inf.*, vol. 13, no. 1, pp. 71–79, Feb. 2017.

[22] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, and A. Rashkovska, "Predictive modeling of PV energy production: How to set up the learning task for a better prediction?" *IEEE Trans. Ind. Inform.*, vol. 13, no. 3, pp. 956–966, Jun. 2017.

[23] X. Zhang, F. Fang, and J. Liu, "Weather-classification-MARS-based photovoltaic power forecasting for energy imbalance market," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, p. 8692–8702, Nov. 2019.

[24] M. Ceci, R. Corizzo, D. Malerba, and A. Rashkovska, "Spatial autocorrelation and entropy for renewable energy forecasting," *Data Min. Knowl. Discovery*, vol. 33, p. 698–729, 2019.

[25] C. Cecati, C. Citro, and P. Siano, "Combined operations of renewable energy systems and responsive demand in a smart grid," *IEEE Trans. Sustain. Energy*, vol. 2, no. 4, pp. 468–476, Oct. 2011.

[26] "Photovoltaic geographical information system (PVGIS)." [Online]. Available: http://re.jrc.ec.europa.eu/pvgis/, Accessed on: Feb. 2, 2019.

[27] "NE ISO pricing reports." [Online]. Available: https://www.iso-ne.com/markets-operations, Accessed on: Feb. 2, 2019.

[28] "NSRDB data viewer." [Online]. Available: https://www.nrel.gov/gis/data-tools.html, Accessed on: Feb. 2, 2019.

[29] A. Ahmad, N. Javaid, M. Guizani, N. Alrajeh, and Z. A. Khan, "An accurate and fast converging short-term load forecasting model for industrial applications in a smart grid," *IEEE Trans. Ind. Inform.*, vol. 13, no. 5, pp. 2587–2596, Oct. 2017.

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 anlytics, 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.