CHAPTER SEVEN

# Advanced machine learning applications to modern power systems

**Cong Feng, Mucun Sun, Morteza Dabbaghjamanesh, Yuanzhi Liu and Jie Zhang**
Department of Mechanical Engineering, The University of Texas at Dallas, Richardson, TX, United States

## Contents



## 7.1 Introduction

Artificial intelligence (AI), the simulation of human intelligence by machines, has brought technological revolutions to the industry and has become part of our life. AI has surpassed human in many fields, including visual recognition, language processing, reading, and playing video games,

*New Technologies for Power System Operation and Analysis*.
DOI: https://doi.org/10.1016/B978-0-12-820168-8.00007-9

and is assisting in autonomous driving, robotic surgery, and legal judgment. In the energy sector, AI is also making profound impacts.

With the deployment of smart grid technologies, power systems have benefited from AI techniques, as shown in Fig. 7.1. The power system data has characteristics of high-volume, high-velocity, and high-variety. For example, sensors such as phasor measurement units (PMUs) take measurements at a millisecond resolution. The advanced metering infrastructure (AMI) in the New York state collects more than 127 TB of consumption data per day [1]. With the big data in power systems, AI provides new solutions to system planning, operation, maintenance, market monitoring, and risk management [2]. For example, reinforcement learning (RL) has been used for power system stability control [3], automatic generation control (AGC) [4], and optimal power flow control [5]. Deep learning has been applied to energy resource assessment [6], injection attack detection [7], and fault diagnosis [8]. A more comprehensive review on AI applications to power systems can be found in Ref. [9].

Among various AI applications in power systems, forecasting is one of the most popular use cases. Forecasting in power systems is to predict the future load, renewable generation output, or electricity and energy price, which are used to assist power system operations at different timescales [10,11]. Therefore forecasting has been widely studied and adopted in power systems. For example, most of the independent system operators (ISO) in the United States have adopted load, wind, and solar power forecasting to assist their system operations [12]. ISO New England utilizes day-ahead forecasts for the dispatch scheduling of generating capacity, reliability analysis, and maintenance planning for the generators [13]. A number of forecasting projects have been or is being conducted to promote renewable energy forecasting, such as Wind Forecast Improvement Project [14], WindView [15], Watt-Sun [16], and SUMMER-GO [17].

Another emerging topic in power systems that relies on AI techniques is system state estimation. Power system state estimation is to retrieve system dynamics, for example, voltage magnitude and phase angles, from available measurements. Traditional methods have challenges in solving state estimation problems with large scale and nonconvexity [18]. Therefore AI techniques have been introduced to help solve state estimation problems through a learning-based optimization manner [19]. For example, a feedforward neural network was used to initialize the Gauss–Newton algorithm to solve the distribution system state estimation in Ref. [20]. An autoencoder was adopted to estimate the voltage magnitude and angle in Ref. [21].

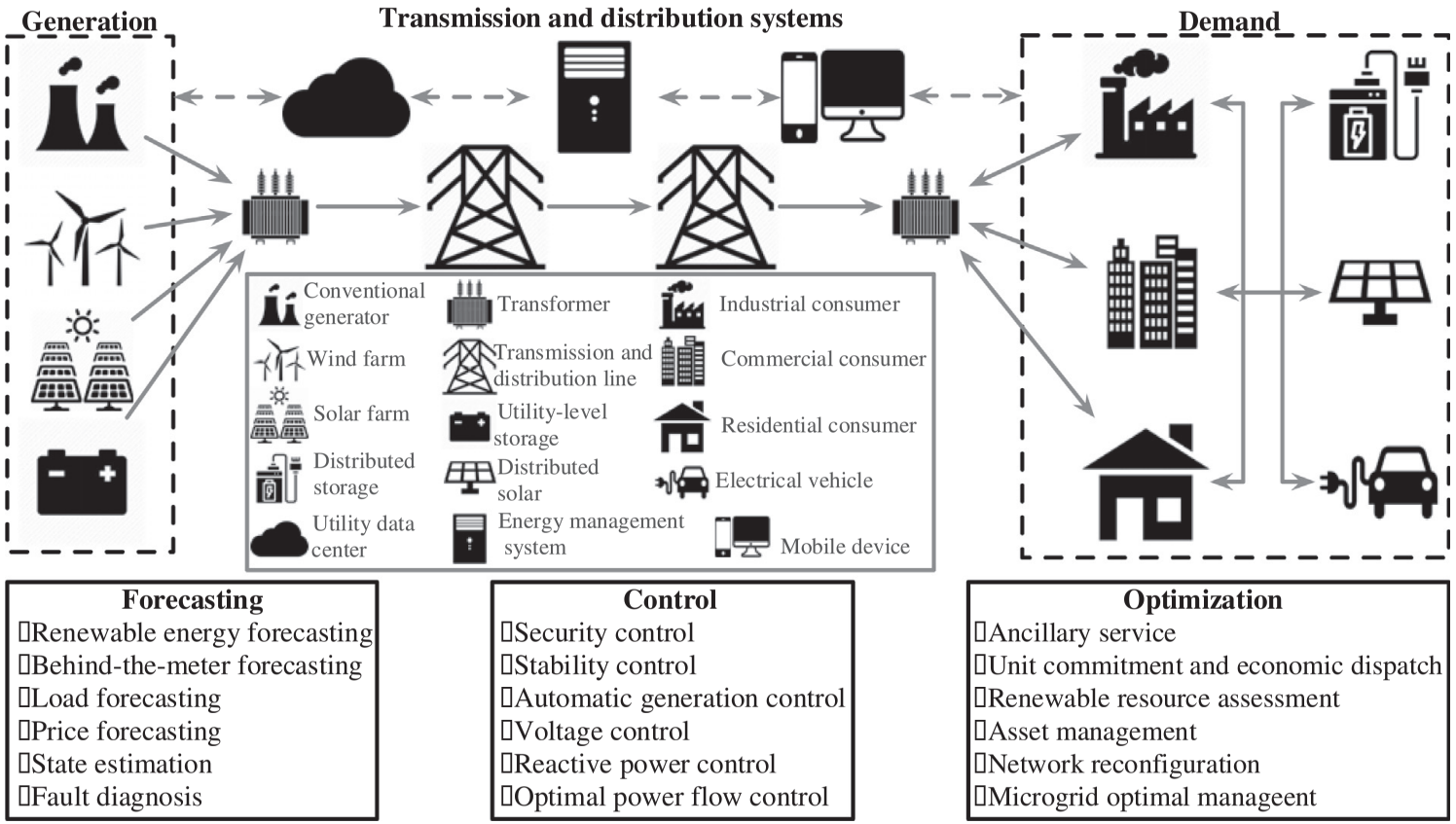


**Figure 7.1** Artificial intelligence applications to smart grid.

In addition, a recurrent neural network (RNN) was trained to estimate the states of the IEEE 57- and 118-bus systems [18]. Bad data detection and cyberattack detection are also recognized to impact the system states and, thus, have also been widely researched. For instance, Hink et al. [22] successfully detected cyberattacks with several different machine learning methods. A support vector machine (SVM) model was used to detect the stealthy false data injection with nearly 95% accuracy [23].

AI has also been applied to building occupancy detection, which is critical to building energy management and building-to-grid integration. The building sector accounts for over 70% of the total electricity consumption in the United States, making the building integration a critical part of the smart grid. The building occupancy information helps building management in several ways, such as occupancy-driven demand response and building energy management. For example, an occupancy-based feedback control algorithm was applied to a heating, ventilation, and air-conditioning system and achieved 29%–80% energy savings [24]. Korkas et al. [25] developed a control optimization method for demand response management in microgrids considering occupancy information and reduced energy costs by 20%. Due to the remarkable benefits of the occupancy information, accurate occupancy detection is recognized as a crucial factor and has received growing attention [26,27].

To demonstrate the AI applications to power systems, this chapter reviews the state-of-the-art machine learning methods, including ensemble learning and deep learning, in renewable energy forecasting, power system network reconfiguration, and smart building occupancy detection.

## 7.2 Modern forecasting technology

This section reviews the state-of-the-art forecasting methods and discusses the details of two types of ensemble learning-based forecasting techniques.

### 7.2.1 Prior research work

Time series forecasting can be categorized based on different criteria, such as forecasting task (e.g., load, wind, solar, and price), time horizon (e.g., short-term, midterm, and long-term), or methods. Fig. 7.2 summarizes a

rough classification of forecasting methods, which are mainly divided into deterministic forecasting and probabilistic forecasting. The former can obtain accurate forecasting results after a specific time horizon, while the latter can provide probabilistic and confidence levels for the uncertainty of desired forecasts.

#### 7.2.1.1 Deterministic forecasting methods

Generally, deterministic forecasting models can be further divided into three categories: statistical [28], intelligent [29], and ensemble models [30]. Statistical methods refer to the utilization of mathematical theory knowledge such as mathematical statistics, probability theory, and stochastic processes. Intelligent models refer to AI models using machine learning and deep learning. Ensemble models refer to the combination of two different algorithms or methods. Ensemble models can combine the merits and characteristics of different methods, which normally perform better than single models [31]. Conventional statistical models include the autoregressive model, the autoregressive moving average model, and the autoregressive integrated moving average model. The most popular machine learning algorithms in renewable energy forecasting are artificial neural networks (ANNs), support vector regression (SVR) model, random forest (RF), and gradient boosting machine (GBM). For example, Feng et al. [32] proposed a short-term solar forecasting method based on sky imaging and pattern recognition. A least squares SVR model was proposed in Ref. [33] to model the nonlinearity of electric load. A GBM model was developed to quantify the dataset diversity for short-term wind forecasting [34], which helps system operational scheduling such as economic dispatch and

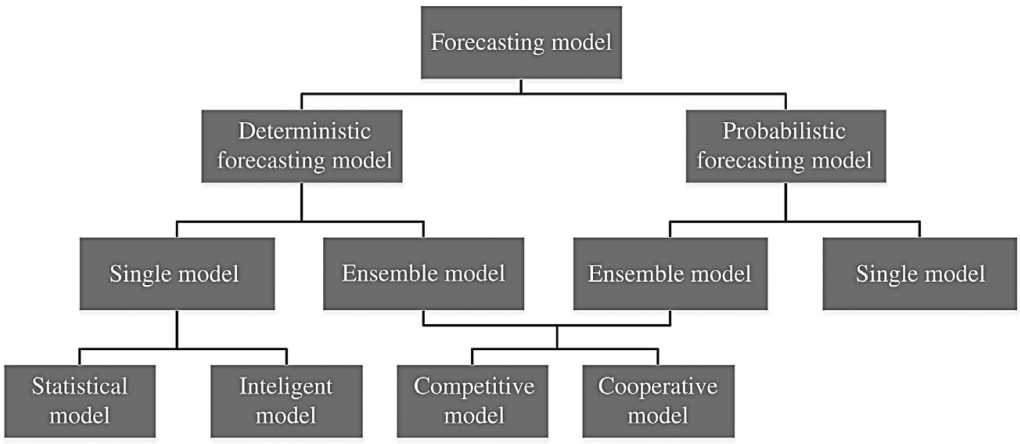


**Figure 7.2** Machine learning–based forecasting methods categorization.

unit commitment. Xia et al. [35] proposed a short-term wind power forecasting method based on neuro-fuzzy network. Results showed that the trained neuro-fuzzy networks could significantly improve the wind forecasting accuracy. More deterministic forecasting models can be found in the latest review papers [36–38].

#### 7.2.1.2 Probabilistic forecasting methods

While deterministic forecasts are critical to power system operations, probabilistic forecasts provide more quantitative uncertainty information associated with desired forecasts and have become extremely important for reliable and economic power system operations. Probabilistic forecasts usually take the form of probability distributions associated with point forecasts, namely, the expectation. Existing methods of constructing predictive distributions can be mainly classified into parametric and nonparametric approaches in terms of distribution shape assumptions. A prior assumption of the predictive distribution shape is made in parametric methods, and the unknown distribution parameters are estimated based on historical data. Parametric approaches generally require low computational cost, while nonparametric approaches estimate the quantiles through a finite number of observations. Landry et al. [39] used GBM for multiple quantile regression to fit each quantile and zone independently and generate probabilistic forecasts. Sun et al. [40] proposed an aggregated probabilistic wind power forecasting method based on spatiotemporal correlation, where Copula was used to model the correlation among wind farms and Gaussian mixture model was used to model the marginal distribution. Zhang and Wang [41] developed a probabilistic forecasting method based on $k$-nearest neighbor ($k$NN) point forecasts through kernel density estimation. Lou et al. [42] used machine learning and multivariable regression to predict diffuse solar irradiance. Wang et al. [43] used deep convolutional neural network (CNN) and wavelet transform to quantify the wind power uncertainties with respect to model misspecification and data noise. Sun et al. [44] developed a probabilistic forecasting method based on pinball loss optimization among different types of predictive distributions.

#### 7.2.1.3 Ensemble learning

The ensemble of individual machine learning models is another efficient way to improve the forecasting accuracy. Methods of constructing

ensemble forecasts can be mainly classified into competitive ensemble and cooperative ensemble methods.

Competitive ensemble methods use induction algorithms with different parameters or initial conditions to build individual models. The final refined ensemble prediction is obtained from pruning and aggregating individual forecasts. Competitive ensemble methods generally require high data and parameter diversity to get different decisions from individual predictors. Therefore competitive ensemble methods usually require high computation cost, and they are usually used in mid-term to long-term forecasting. Bagging and boosting are two commonly used competitive ensemble methods. For example, to better account for the performance of weak models, ensemble forecasting approaches based on adaptive boosting (i.e., assign smaller weights to the models with larger errors) are used in Ref. [45]. Feng et al. [11,46] proposed a machine learning–based multi-model forecasting framework that consists of an ensemble of four single-machine learning algorithms with various kernels to generate deterministic wind forecasts and solar forecasts. Machine learning–based competitive ensemble learning was also used in solar forecasting [47] and load forecasting [48].

For cooperative ensemble methods, the dataset is divided into several subdatasets and each subdataset is forecast separately, and the final forecasts are obtained by aggregating all the subforecasts. ANN-based autoregressive integrated moving average [49] and generalized autoregressive conditional heteroskedasticity-based autoregressive integrated moving average [50] are the two commonly used cooperative ensemble methods that combine suitable models for linear and nonlinear time series. Ye et al. [51] developed an AdaBoost-based empirical mode decomposition (EMD) ANN to generate ensemble wind speed forecasts. Wang et al. [52] developed a cooperative ensemble method to generate wind speed forecasts, which improved EMD through decomposing the original data into more stationary signals with different frequencies. These stationary signals were considered as different inputs to a genetic algorithm and backpropagation-based neural network, and the final forecast was the aggregation of each single prediction.

In addition to wind speed forecasting, ensemble methods have also been applied to probabilistic wind power forecasting and probabilistic solar power forecasting. For example, Lin et al. [53] combined multiple probabilistic forecasting models based on sparse Bayesian learning, kernel density estimation, and beta distribution estimation. The weight

parameters of the multimodel ensemble were solved by an expectation maximizing algorithm and continuous ranked probability score optimization. Kim and Hur [54] developed an enhanced ensemble method for probabilistic wind power forecasting. The wind speed spatial ensemble was built by using correlation-based weight and kriging models, and the temporal ensemble was built through an average ensemble of three models (i.e., an exogenous variable model, a polynominal regression model, and an analog ensemble model).

## 7.2.2 Ensemble learning forecasting methodologies

Both competitive and cooperative ensemble learning rely on single models. Therefore single-machine learning algorithms, including ANN, SVR, GBM, and RF, are first introduced in this section. Then, two wind forecasting methodologies based on competitive or cooperative ensemble learning are described.

### *7.2.2.1 Single-machine learning algorithm models*

ANN is a popular machine learning algorithm in speech recognition, target tracking, signal analysis, and nonlinear regression problems (such as time series forecasting). ANN mimics the action of the human neurons, and each neuron is a weighted sum of its inputs and is connected to the neurons in the next layer. A weight decides how much influence the input will have on the output. The ANN architecture contains one input layer, one or more hidden layer(s), and one output layer. The output signal of ANN can be either 0, 1, or any real value between 0 and 1 depending on whether we are dealing with "binary" or "real valued" artificial neurons. The configuration of the ANN model needs to be well designed to avoid overfitting issues. ANN can be classified into different types with different activation functions and learning algorithms. Sigmoid and hyperbolic tangent are two commonly used activation functions. Deep learning is also a configuration of ANN, where multiple hidden layers are built. The mathematical description of ANN is expressed as:

$$y_i^{(n)} = f\left(\sum_{j=1}^{N} w_{ij}^{(n,n-1)} y_j^{(n-1)} + \theta_i^n\right) \tag{7.1}$$

where $i$ is a neuron of the $n$th layer, $w_{ij}$ is the weight from the neuron $j$ in the layer $(n-1)$ to the neuron $i$ in layer $n$, and $\theta_i^n$ is the threshold of the neuron $i$ in layer $n$.

SVM is originally a supervised linear classifier proposed by Cortes and Vapnik [55]. As one of the most popular classification methods, SVM has been applied in text categorization, image classification, and other recognition tasks. When dealing with linearly inseparable data, nonlinear mapping-based kernel methods, $\mathrm{K}(x):R^{n} \longrightarrow R^{n_h}$, are used to map the nonlinear data into the high-dimensional feature space. Then, a linear hyper plane can be found by maximizing the distance between support vectors and the hyper plane. The SVM algorithm can also be applied in regression problems, which is called SVR. However, the compute and storage requirements increase significantly with the data dimension. The hyper plane function, also called the SVR function, is described as [56]:

$$f(x) = \omega^{T}\mathrm{K}(x) + b \tag{7.2}$$

where $\omega$ and $b$ are variables solved by minimizing the empirical risk, which is given by:

$$R(f) = \frac{1}{n}\sum_{i=1}^{n}\Theta(y_i, f(x)) \tag{7.3}$$

where $\Theta_{\varepsilon}(y_i, f)$ is the $\varepsilon$-insensitive loss function, expressed as:

$$\Theta_{\varepsilon}(y_i, f) = \begin{cases} \|f - y\| - \varepsilon, \text{if } \|f - y\| \geq \varepsilon \\ 0, \text{otherwise} \end{cases} \tag{7.4}$$

Then the optimal hyper plane can be found by solving the inequality-constrained quadratic optimization problem.

GBM is a highly customizable learning algorithm, which is widely used in the regression and classification fields. GBM model relies on the combination of "weak learners" to create an accurate learner and, therefore, is able to generate both deterministic and probabilistic results in time series forecasting. The combination is achieved by adding the weighted base learner to the previous model iteratively. The principle of GBM is illustrated by the pseudo-code in Algorithm 1. In each iteration the negative gradient of the chosen loss function is calculated and used to estimate the split variables $a$ by Eqs. (7.5) and (7.6). Then the multiplier $\beta$ is optimized by Eq. (7.7). The weak learner $\beta h(x;\ a)$ is added to the previous model, where $h(x;a)$ is a learning function.

RF is another supervised ensemble learning method that consists of many single classification and regression trees (CARTs):

**Algorithm 1 Gradient boosting machine**

1 Initialize $f_0(x)$ to be a constant, $f_0(x) = \text{argmin}_\rho \sum_{i=1}^{n} \Psi(y_i, \rho)$

2 for $i = 1$ to M do

3 Compute the negative gradient of the loss function:

$$\overline{y_i} = -\left[\frac{\partial \Psi(y_i, F(x_i))}{\partial F(x_i)}\right]_{f(x)} = f_{i-1}(x), \quad i = \{1, 2, \ldots, n\} \tag{7.5}$$

4 Fit a model to $\overline{y}$ by least squares to get $a_t$:

$$a_t = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} [\overline{y_i} - \beta h(x_i, a)]^2 \tag{7.6}$$

5 Calculate $\beta_t$ by:

$$\beta_t = \arg\min_{\beta} \sum_{i=1}^{n} \Psi\left(y_i, f_{t-1}(x_i) + \beta h(x_i, a_t)\right) \tag{7.7}$$

6 Update the model by:

$$f_t(x) = f_{t-1}(x) + \beta_t h(x; a_t) \tag{7.8}$$

7 end for

8 Output $\hat{f}(x) = f_T(x)$

$$T = \left\{t\left(X, s_{\Lambda_1}\right), t\left(X, s_{\Lambda_2}\right), \ldots, t\left(X, s_{\Lambda_n}\right)\right\} \tag{7.9}$$

where $T$ is the set of CARTs, $t$ is a single CART, $X$ is the input to the RF model, and $s_{\Lambda_i}$ is the random vector to extract bootstrap samples that are determined by the bagging algorithm. The robustness of the RF model is enhanced by the randomness of the bagging algorithm and the best splits search process. Since RF is a combination of various regressions, the model is generally free from overfitting [57].

#### *7.2.2.2 Competitive ensemble learning*

A competitive ensemble methodology is developed for short-term wind forecasting with both deterministic and probabilistic forecasts, which is called the machine learning–based multi-model forecasting framework (M3), as shown in Fig. 7.3. The M3 deterministic model has two layers (note that this is different from ANN layers). The first layer consists of several machine learning models, that is, ANN, SVM, GBM, and RF, which are built based on the historical data. These models forecast wind

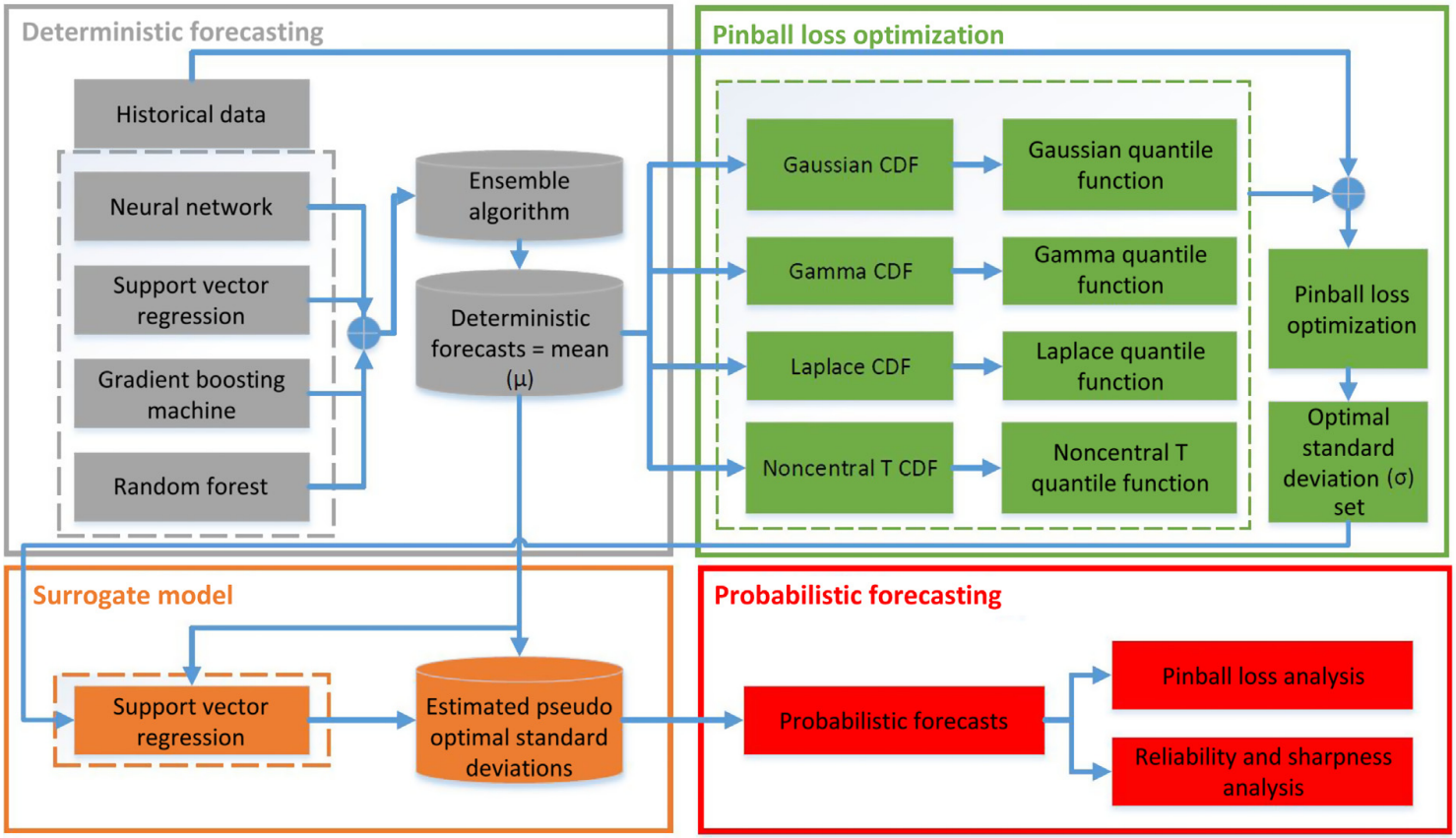


**Figure 7.3** Competitive ensemble learning framework based on the M3 model.

speed or wind power as the output. A blending model is developed in the second layer to combine the forecasts produced by different algorithms from the first layer, and to generate both deterministic and probabilistic forecasts. This blending model is expected to take advantage of different algorithms by canceling or smoothing the local forecasting errors. The mathematical description is shown as:

$$y_i = f_i(x_1, x_2, \ldots, x_p) \tag{7.10}$$

$$\hat{y} = \Phi(y_1, y_2, \ldots, y_m) \tag{7.11}$$

where $f_i(^*)$ is the $i$th algorithm and $y_i$ is the wind speed forecasted by $f_i(^*)$. $\hat{y}$ is the second-layer blending algorithm.

The M3 deterministic forecasts can also be transferred to probabilistic forecasts. Specifically, after obtaining deterministic forecasts, a set of unknown parameters in the predictive distribution are determined by minimizing the pinball loss that is an evaluation metric of probabilistic forecasts. Note that the optimal distribution parameters are adaptively and dynamically updated based on the deterministic forecast value at each time stamp. The optimal adaptive predictive distribution parameters are first determined offline with the historical training data. Then a surrogate model is developed to represent the optimized distribution parameter as a function of the deterministic forecast. At the real-time forecasting stage the surrogate model is used together with deterministic forecasts to adaptively predict the unknown distribution parameters and thereby generate probabilistic forecasts.

#### 7.2.2.3 Cooperative ensemble learning

In addition to deterministic forecasts, the ensemble methods could also be applied to probabilistic forecasts directly. A cooperative ensemble methodology of probabilistic wind power forecasts from different type of predictive distributions is introduced in this section, and the overall framework is illustrated in Fig. 7.4.

In the deterministic forecasting stage, instead of integrating multiple single-machine learning algorithms together to get a refined forecast, a Q-learning-enhanced deterministic forecasting method [58] is adopted to select the best forecasting model from a pool of state-of-the-art machine learning–based forecasting models (i.e., ANN, SVR, GBM, and RF) at each time step. To be more specific, the developed method trains Q-learning agents based on the rewards of transferring from the current model to the next model. For example, a Q-learning agent will receive a

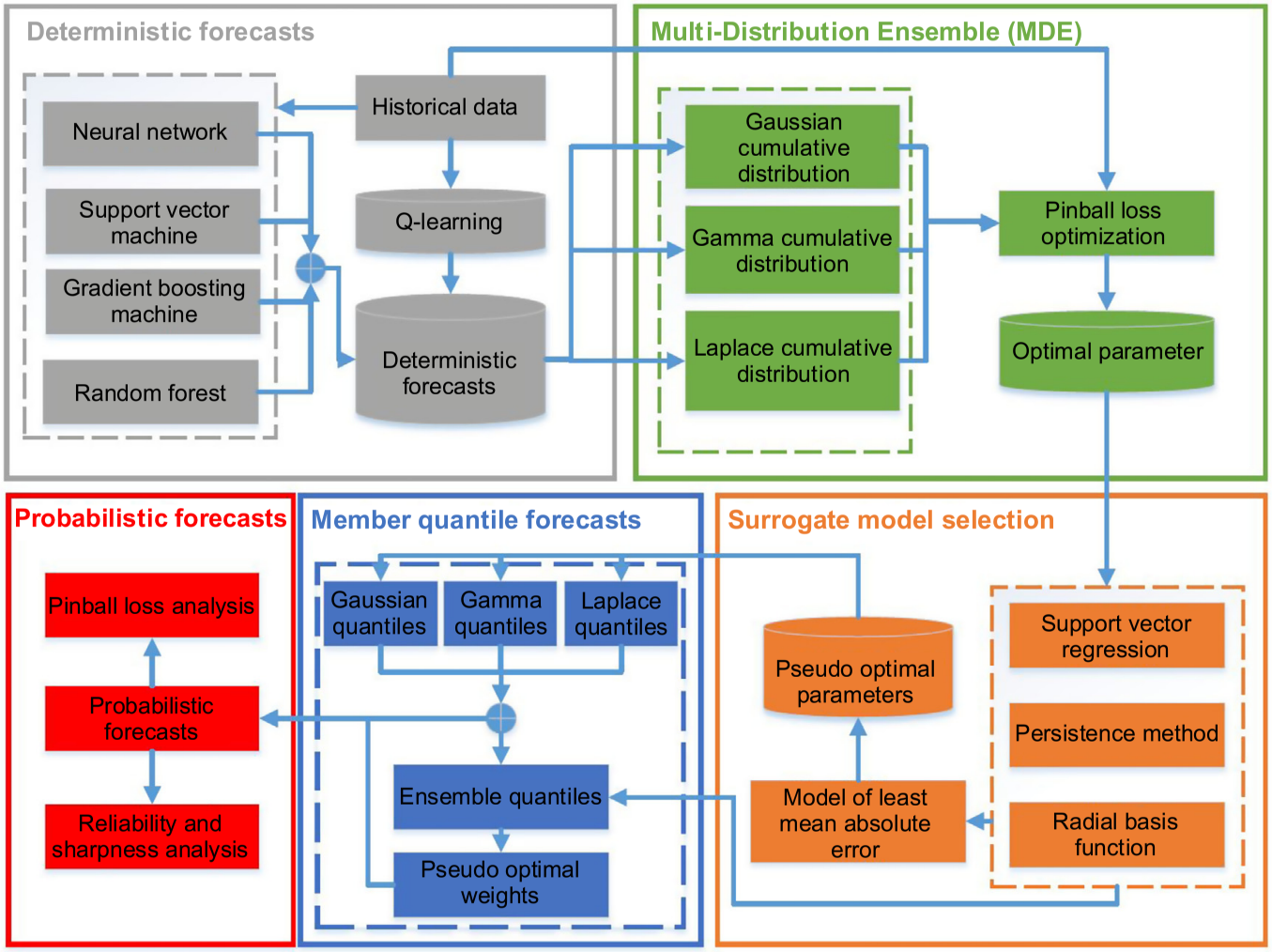


**Figure 7.4** The overall framework of the cooperative ensemble probabilistic forecasting.

reward by transferring from the current forecasting model $M_i$ to the next forecasting model $M_j$ in each training step, from which the Q-learning agent will learn the optimal policy of the model selection. Then, this optimal policy will be applied to select the best model for forecasting in the next step based on the current model in the forecasting stage. The dynamic model selection process is expressed as:

$$S = \{s\} = \{s_1, s_2, \ldots, s_I\} \tag{7.12}$$

$$A = \{a\} = a_1, a_2, \ldots, a_I \tag{7.13}$$

$$R^t\left(s_i, a_j\right) = \text{ranking}\left(M_i\right) - \text{ranking}(M_j) \tag{7.14}$$

$$Q^{e+1}(s_e, a_e) = (1-\alpha)Q^e(s^e, a^e) + \alpha\left[R^e(s^e, a^e) + \gamma \max Q^e\left(s^{e+1}, a\right)\right] \tag{7.15}$$

where $S$, $A$, $R$, and $Q$ are state space, action space, reward space, and Q-table in the dynamic model selection Markov Decision Process, respectively. The parameters $s$ and $a$ are possible state and action, respectively. The parameter $I$ is the number of models ($M$) in the model pool, $e$ is the episode index with the maximum of 100, $\alpha = 0.1$ is the learning rate that controls the aggressiveness of learning, and $\gamma = 0.8$ is a discount factor that weights the future reward. The reward function is defined as the model performance improvement, which ensures the effective and efficient convergence of Q-learning. More details of this Q-learning-based dynamic forecasting model selection can be found in Ref. [58].

In the probabilistic forecasting stage, individual probabilistic forecasts are generated by using each single predictive distribution based on the training dataset [44]. The unknown parameters (i.e., standard deviations) of each predictive distribution are optimized. A weight parameter is assigned to quantile forecasts from each individual model, and these weight parameters are optimized again by minimizing the pinball loss. Then a surrogate model is developed to represent each optimal weight as a function of the deterministic forecast. During online forecasting a set of pseudo-optimal parameters of the ensemble model is estimated by the surrogate model and deterministic forecasts. Finally, the method with the minimum pinball loss is chosen to produce the final ensemble probabilistic forecasts.

### 7.2.3 Forecasting results

Performance of the single-algorithm and ensemble machine learning models is evaluated in this section for both deterministic and probabilistic

wind speed/power forecasting. The ensemble deterministic and multi-model probabilistic forecasting models are tested on wind speed forecasting. The Q-learning–enhanced deterministic and ensemble probabilistic forecasting models are tested on wind power forecasting. Two evaluation metrics are utilized to evaluate the deterministic forecasting accuracy in both case studies: the normalized mean absolute error (nMAE) and the normalized root mean square error (nRMSE):

$$\mathrm{nMAE} = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{\hat{x}_i - x_i}{x_{\max}} \right| \tag{7.16}$$

$$\mathrm{nRMSE} = \frac{1}{x_{\max}} \sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n}} \tag{7.17}$$

where $\hat{x}_i$ is the forecast value, $x_i$ is the actual value, $x_{\max}$ is the maximum actual value, and $n$ is the sample size.

Pinball loss, one of the most popular metrics, is used to evaluate the performance of probabilistic forecasting in both case studies. Pinball loss is a function of observations and quantiles of a forecast distribution. A smaller pinball loss value indicates better probabilistic forecasting. The pinball loss value of a certain quantile $L_m$ is expressed as:

$$L_m(q_m, x_i) = \begin{cases} \left(1 - \dfrac{m}{100}\right) \times (q_m - x_i), \& x_i < q_m \\ \dfrac{m}{100} \times (x_i - q_m), \& x_i \geq q_m \end{cases} \tag{7.18}$$

where $x_i$ represents the $i$th observation, $m$ represents a quantile percentage from 1 to 99, and $q_m$ represents the predicted quantile. For a given $m$ percentage the quantile $q_m$ represents the value of a random variable whose CDF is $m$ percentage.

#### 7.2.3.1 Case study I: wind speed forecasting based on competitive ensemble learning

Both the single-algorithm models and ensemble algorithm are applied to the wind speed data collected near hub height with a 1-hour resolution at eight sites listed in Table 7.1. For all the eight locations the first 2/3 of data are used as training data, in which the first 11/12 is used to train ensemble algorithm and the remaining 1/12 of the training data is used to build the surrogate model of the optimal standard deviation. The accuracy

of the forecasts is evaluated by the remaining 1/3 of data. The nMAE and nRMSE are compared in Tables 7.2 and 7.3, respectively.

The multi-model framework includes multiple individual models in the first layer and several models in the second layer. Different algorithms are tested in both layers, which include (1) three SVR models with linear (SVR_l), polynomial (SVR_p), and radial base (SVR_r) kernels; (2) five ANN models with different numbers of hidden layers ($n_l$), neurons in each layer ($n_o$), and weight decay parameter ($n_d$) values, and the selected models employ the feed-forward−backpropagation learning function and

**Table 7.1** Data duration at selected sites.

| Name | Data duration | Height (m) |
|---|---|---|
| Boulder_NWTC (C1) | 2009-01-02 to 2012-12-31 | 80 |
| Megler (C2) | 2010-11-03 to 2012-11-01 | 53.3 |
| CedarCreek_H06 (C3) | 2009-01-02 to 2012-12-31 | 69 |
| Goodnoe_Hills (C4) | 2007-01-01 to 2009-12-31 | 59.4 |
| Bovina50 (C5) | 2010-10-10 to 2012-10-08 | 50 |
| Bovina100 (C6) | 2010-03-03 to 2012-03-01 | 100 |
| CapeMay (C7) | 2007-09-26 to 2009-09-24 | 100 |
| Cochran (C8) | 2008-06-30 to 2011-06-29 | 70 |

*Note*: The case notations (C1−C8) are different from Sections 7.2.3.2 and 7.4.3.

**Table 7.2** The normalized mean absolute error (nMAE) (%) of 1HA forecasts.

| Method | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 |
|---|---|---|---|---|---|---|---|---|
| SVR_r | 5.101 | 3.114 | 6.229 | 3.799 | 5.145 | 4.554 | 3.662 | 4.014 |
| SVR_l | 4.765 | 2.886 | 3.927 | 3.718 | 4.891 | 4.466 | 3.449 | 3.984 |
| SVR_p | 4.772 | 2.919 | 4.267 | 3.734 | 4.913 | 4.572 | 3.553 | 4.009 |
| ANN1 | 4.793 | 2.921 | 4.155 | 3.738 | 4.936 | 4.671 | 3.717 | 4.007 |
| ANN2 | 4.789 | 2.938 | 4.042 | 3.735 | 4.939 | 4.536 | 3.560 | 4.012 |
| ANN3 | 4.817 | 2.927 | 4.096 | 3.738 | 4.932 | 4.494 | 3.502 | 4.017 |
| ANN4 | 4.792 | 2.906 | 4.022 | 3.735 | 4.924 | 4.481 | 3.500 | 4.005 |
| ANN5 | 4.793 | 2.902 | 3.859 | 3.727 | 4.899 | 4.487 | 3.480 | 4.006 |
| GBM1 | 4.822 | 2.945 | 4.468 | 3.739 | 4.961 | 4.479 | 3.562 | 4.022 |
| GBM2 | 4.808 | 2.941 | 4.474 | 3.736 | 4.963 | 4.478 | 3.550 | 4.020 |
| GBM3 | 4.806 | 2.936 | 4.730 | 3.768 | 4.969 | 4.491 | 3.504 | 4.002 |
| GBM4 | 4.845 | 2.946 | 4.348 | 3.754 | 4.974 | 4.544 | 3.554 | 4.023 |
| RF1 | 4.965 | 3.060 | 4.207 | 3.883 | 5.115 | 4.703 | 3.715 | 4.159 |
| RF2 | 4.920 | 3.012 | 4.221 | 3.852 | 5.057 | 4.637 | 3.659 | 4.110 |
| M3 | **4.623** | **2.712** | **3.731** | **3.654** | **4.683** | **4.256** | **3.223** | **3.871** |

*Note*: The smallest nMAEs among all the models are in bold.

**Table 7.3** The normalized root mean square error (nRMSE) (%) of 1HA forecasts.

| Method | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 |
|---|---|---|---|---|---|---|---|---|
| SVR_r | 8.039 | 4.883 | 11.769 | 5.318 | 7.166 | 6.363 | 5.383 | 5.477 |
| SVR_l | 6.954 | 3.993 | 5.638 | 5.095 | 6.539 | 6.224 | 4.813 | 5.401 |
| SVR_p | 6.953 | 4.011 | 6.381 | 5.109 | 6.545 | 6.350 | 4.946 | 5.423 |
| ANN1 | 6.907 | 4.019 | 6.250 | 5.123 | 6.585 | 6.364 | 5.065 | 5.409 |
| ANN2 | 6.908 | 4.038 | 5.974 | 5.105 | 6.589 | 6.260 | 4.930 | 5.408 |
| ANN3 | 6.902 | 4.026 | 6.116 | 5.123 | 6.579 | 6.223 | 4.868 | 5.402 |
| ANN4 | 6.930 | 4.024 | 5.913 | 5.124 | 6.564 | 6.225 | 4.888 | 5.411 |
| ANN5 | 6.919 | 4.005 | 5.483 | 5.096 | 6.529 | 6.207 | 4.826 | 5.402 |
| GBM1 | 6.969 | 4.105 | 7.197 | 5.100 | 6.605 | 6.225 | 4.942 | 5.413 |
| GBM2 | 7.016 | 4.102 | 7.225 | 5.103 | 6.613 | 6.237 | 4.936 | 5.419 |
| GBM3 | 7.059 | 4.121 | 7.886 | 5.217 | 6.654 | 6.258 | 4.916 | 5.421 |
| GBM4 | 7.011 | 4.116 | 6.898 | 5.112 | 6.620 | 6.291 | 4.955 | 5.426 |
| RF1 | 7.220 | 4.238 | 6.248 | 5.261 | 6.788 | 6.463 | 5.154 | 5.585 |
| RF2 | 7.141 | 4.159 | 6.416 | 5.224 | 6.715 | 6.388 | 5.067 | 5.526 |
| M3 | **6.720** | **3.988** | **5.524** | **4.987** | **6.341** | **6.089** | **4.760** | **5.101** |

*Note*: The smallest nRMSEs among all the models are in bold.

sigmoid activation function; (3) four GBM models based on different loss functions (Gaussian and Laplacian) and parameters, that is, number of trees, learning rate ($\lambda$), maximum depth of variable interactions, and minimum number of observations in the terminal nodes; and (4) two RF models with different numbers of variables that are randomly sampled as candidates at each split. In the probabilistic forecasting stage, four widely used predictive distribution types (i.e., Gaussian, Gamma, Laplace, and noncentral *t*) are considered to model the possible shapes of the predictive distribution. The one with the lowest pinball loss is chosen as the final predictive distribution.

It is seen that none of the single models performs better than the ensemble method. The ensemble models have improved the forecasting accuracy of the component models by up to 12.9% based on nMAE and 16.4% based on nRMSE. For the blending algorithms the models with nonlinear blending algorithms have better performance than the models with linear blending algorithms. This shows that the forecasts produced from the first-layer models exhibit a nonlinear relationship with the actual wind speed.

Fig. 7.5 provides an example of the deterministic forecasts along with the confidence intervals in the form of interval plot at the C2 site from 2012-02-01 to 2012-02-04. The colors of the intervals fade with the increasing confidence level, ranging from 10% to 90% in a 10% increment. The intervals are symmetric around the deterministic forecasting curves with a changing width. When the wind speed fluctuates within a

small range, the confidence bands are narrow. When there is a significant ramp, the uncertainty of the forecasts is increased and the bands tend to be broader, as shown by hours 50–65. This further proves the necessity of probabilistic forecasting.

The pinball loss values of the eight selected sites with different predictive distributions are summarized in Table 7.4. The sum of pinball loss is averaged over all quantiles from 1% to 99% and normalized by the maximum wind speed at each site. A lower pinball loss score indicates a better probabilistic forecast. Table 7.4 shows that the M3-Laplace with pinball loss optimization has the smallest pinball loss value at all locations. The M3-Laplace model has reduced the pinball loss by up to 35% compared to the M3 forecasts with other predictive distributions [i.e., Gaussian, Gamma, and Noncentral T (nCT)]. Therefore the Laplace distribution is finally chosen to generate probabilistic wind speed forecasts. Note that the models of M3-Gaussian, M3-Gamma, and M3-Laplace perform similarly,

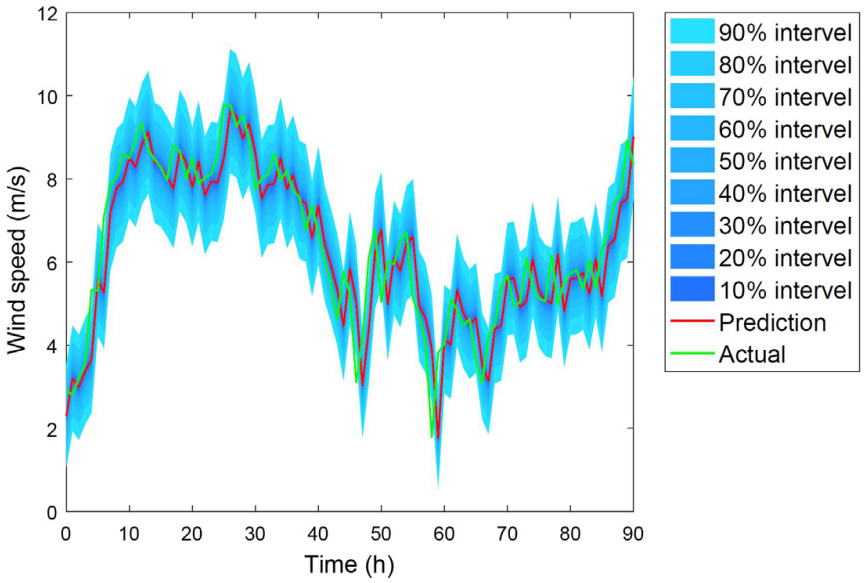


**Figure 7.5** Multi-model probabilistic forecasts at the C2 site.

**Table 7.4** Normalized averaged sum of pinball loss.

| Method | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 |
|---|---|---|---|---|---|---|---|---|
| M3-Gaussian | 1.74 | 1.26 | 1.44 | 1.36 | 1.86 | 1.69 | 1.28 | 1.59 |
| M3-Gamma | 1.74 | 1.26 | 1.44 | 1.36 | 1.87 | 1.69 | 1.27 | 1.58 |
| M3-Laplace | **1.72** | **1.25** | **1.43** | **1.35** | **1.85** | **1.63** | **1.26** | **1.57** |
| M3-nCT | 1.74 | 1.81 | 2.20 | 2.21 | 2.68 | 3.41 | 2.56 | 2.86 |

*Note*: The smallest normalized averaged sum of pinball loss at each location is in boldface.

which indicates that the optimization can help achieve better accuracies with different predictive distribution types.

#### 7.2.3.2 Case study II: wind power forecasting based on cooperative ensemble learning

The Q-learning enhanced deterministic forecasting algorithm is applied to the wind power data collected from the WIND Toolkit with 1-hour resolution, which includes seven sites (as shown in Table 7.5) with meteorological information (e.g., wind direction, wind speed, air temperature, surface air pressure, and density at hub height). For all the locations the first 3/4 of the data is used as training data, in which the first 11/12 is used to train the deterministic forecast models and the remaining 1/12 of the training data is used to build the surrogate models of the optimal standard deviations and weight parameters. The effectiveness of the forecasts is validated by the remaining 1/4 of the data. The nMAE and nRMSE of 1HA (hour-ahead) wind power forecasting are listed in Table 7.6. It is shown that the 1HA nMAE and nRMSE from Q-learning model are in the ranges of 5%–8% and 8%–13%, respectively. To show the effectiveness of the Q-learning enhanced deterministic forecasting, the persistence method is used as a baseline. It is seen from Table 7.6 that overall the Q-learning performs better than the persistence method.

**Table 7.5** Data summary of the selected seven WIND Toolkit sites.

| Site ID (case notation) | Latitude | Longitude | Capacity (MW) | State |
|---|---|---|---|---|
| 4816 (C1) | 29.38 | − 100.37 | 80 | TX |
| 8979 (C2) | 31.53 | − 95.62 | 53.3 | TX |
| 10069 (C3) | 32.31 | − 98.26 | 69 | TX |
| 10526 (C4) | 32.44 | − 100.55 | 59.4 | TX |
| 1342 (C5) | 27.12 | − 97.86 | 50 | TX |
| 2061 (C6) | 27.95 | − 97.40 | 100 | TX |
| 9572 (C7) | 31.99 | − 100.18 | 100 | TX |

*Note*: The case notations (C1–C7) are different from Sections 7.2.3.1 and 7.4.3.

**Table 7.6** The normalized mean absolute error (nMAE) (%) and the normalized root mean square error (nRMSE) [%] of 1HA forecasts.

| Model | Metric | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
|---|---|---|---|---|---|---|---|---|
| Q-learning | nMAE | 6.63 | 6.85 | 6.70 | 6.74 | 6.67 | 5.38 | 7.76 |
| | nRMSE | 10.55 | 10.93 | 11.04 | 10.66 | 9.93 | 8.37 | 11.93 |
| Persistence | nMAE | 7.14 | 7.18 | 6.97 | 7.11 | 7.12 | 5.74 | 8.06 |
| | nRMSE | 11.38 | 11.71 | 11.82 | 11.50 | 10.99 | 8.93 | 12.63 |

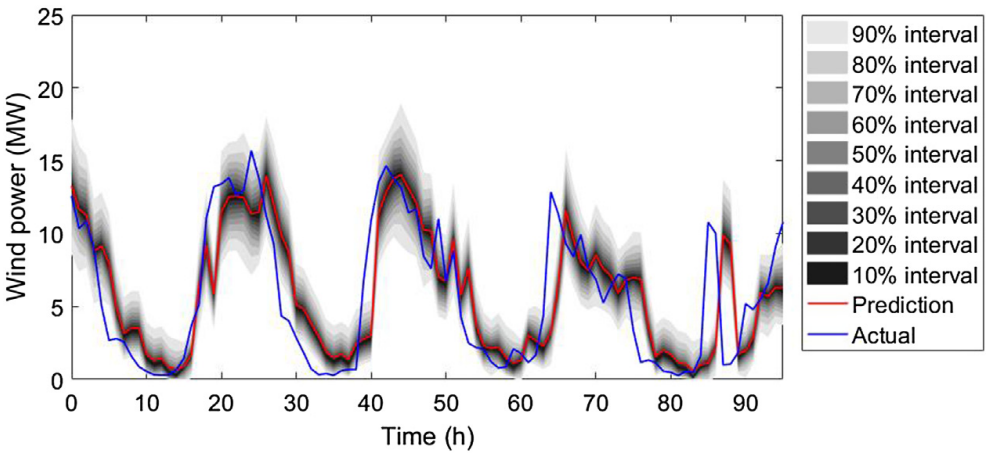


**Figure 7.6** Cooperative ensemble probabilistic forecasts at the C2 site.

**Table 7.7** Normalized averaged sum of pinball loss.

| Method | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
|---|---|---|---|---|---|---|---|
| Q-learning-cooperative | 2.23 | 2.58 | 2.42 | 3.01 | 3.05 | 1.98 | 3.67 |
| Q-learning without ensemble | 2.46 | 2.83 | 2.67 | 3.38 | 3.42 | 2.30 | 3.94 |

To better visualize the ensemble forecasting model, Fig. 7.6 provides an example of the deterministic forecasts along with the confidence intervals in the form of interval plot at the C2 site from 2012-07-20 to 2012-07-24. The colors of the intervals fade with the increasing confidence level, ranging from 10% to 90% in a 10% increment. The intervals are symmetric around the deterministic forecasting curves with a changing width. Fig. 7.6 shows that the prediction intervals of the ensemble forecasting model are narrow, which show less uncertainty in the probabilistic forecasts.

The pinball loss values of the seven selected sites with different predictive distributions are summarized in Table 7.7. Results show that the Q-learning-based ensemble probabilistic forecasting method has reduced the pinball loss by up to 16.1% compared with single probabilistic forecasting methods.

## 7.3 Machine learning–based control and optimization

The powerful learning abilities also enable machine learning models to solve complex control and optimization problems in power systems.

This section reviews the popular applications of machine learning in power system control and optimization. Specifically, a network reconfiguration optimization problem is solved by AI to demonstrate the "learn to optimize" capability of machine learning models.

## 7.3.1 Prior research work

### *7.3.1.1 Machine learning–based control*

Due to the increasing complexity and uncertainty brought by renewable energy, distributed energy resources, and varying loads, it is of significant importance for the electric power system to employ advanced control strategies to keep the system working reliably and efficiently. Studies have shown that machine learning–based control strategies are capable of addressing the high-dimensional complex nonlinear control challenges, and it could be more effective when properly combined together with conventional mathematical approaches [59,60].

In terms of power system stability, extensive studies have employed neural network to design power system stabilizer (PSS), and controller for AGC or frequency control [61]. For instance, a self-tuning PSS based on ANN was proposed by Segal et al. [62]. In this approach, ANN was introduced for tuning the conventional PSS parameters in real time, showing that the dynamic performance of the system with the ANN-based PSS is robust over a wide range of loading conditions and equivalent reactance. To improve the transient stability of power systems under different operating conditions and parametric uncertainties, Senjyu and Fujita [63] proposed an RNN stabilization controller for both automatic voltage regulators and the governor, where the weights of the controller are adjusted online. The proposed approach was applied to a single-machine infinite-bus system and showed good damping characteristics over a wide range of operating conditions. However, due to the complexity of large-scale power systems, design of intelligent controllers–based ANN requires large training time and a large number of neurons. These drawbacks motivated Chaturvedi et al. [64] to utilize generalized neurons (GNs) to develop a GN-based adaptive PSS. Given the advantages of self-optimizing adaptive control and the quick response of the GN, the proposed GN-based PSS can provide good damping of the power system over a wide operating range and significantly improved the dynamic performance. To stabilize the system after severe disturbances and mitigate the oscillations afterward, Hadidi and Jeyasurya [65] employed a Q-learning RL algorithm to develop a real-time closed-loop

wide-area decentralized PSSs. The proposed algorithm proved that the stability boundary of the system could be extended without losing any generator or load area.

Due to the deviation between system load and power generation, there exist frequency fluctuations in the power system, which requires a supplementary control strategy to restore the system frequency to the nominal values. For example, Zeynelgil et al. [66] proposed an ANN controller with the backpropagation through time algorithm to study AGC in a four-area interconnected power system. Chaturvedi et al. [67] proposed a generalized NN-based decentralized controller to control the frequency deviations in power systems. Other modified NN algorithms include dynamic NN [68] and radial base function neural network [69]. Moreover, RL algorithms, that is, model-free Q-learning [70], hierarchically correlated equilibrium Q-learning [71], and deep distributed recurrent Q-networks-action discovery [72], have also been employed to develop frequency controllers for different conditions and purposes.

#### 7.3.1.2 Machine learning–based optimization

For a data-driven nonlinear optimization problem, the first step in general is to establish a mathematical regression model as the objective function in terms of the design variables based on the experimental or collected data. Since the physical or explicit relationships are unable to be observed directly or remain unknown, the modeling process is referred to as black-box. Machine learning (i.e., surrogate prediction model) approaches are widely employed to solve the black-box problem, such as ANN, kriging/Gaussian process (GP) regression, SVM, radial basis functions, and polynomial response surface. By cross-validation the best appropriate models are selected to build up the black-box model. The next step of the optimization process is to apply a gradient-based or heuristic algorithm to solve the problem.

The machine learning–based optimization has also been extensively employed in power system studies. For example, Lucifredi et al. [73] compared the kriging and ANN statistical modeling techniques with a conventional linear multiple regression method for hydroelectric power system maintenance prediction. Pan and Das [74] proposed to use a fractional order control strategy for a microgrid, in which the controller parameters were tuned with a global optimization algorithm to meet system performance specifications. In this research, a kriging-based surrogate modeling technique integrated with GA was employed to alleviate the

issue of expensive objective function evaluation for the optimization-based controller tuning. Luh et al. [75] utilized augmented Lagrangian relaxation to form and solve the market clearing prices subproblems by using a surrogate-based optimization framework.

## 7.3.2 A Machine learning–based network reconfiguration methodology

### *7.3.2.1 Literature review on network reconfiguration*

Over the past decades, power grids have become more prone to be overloaded due to the disproportionate increasing of the load demand and generation units (both dispatchable and nondispatchable units) [76]. Hence, the reconfiguration technique has been proposed to prevent overloading and provide an efficient power dispatch, especially in the emergency conditions when the line current approaches its maximum ampacity [77]. By definition the reconfiguration is the process of changing the topology of the power network by some prelocated sectionalizing and tie switches that can significantly improve the grid reliability [77], voltage profile [78], line loss [79], and load balance [80]. Up to now, many heuristics and mathematical algorithms have been used to solve the reconfigurable power grids, for example, collective decision-based algorithm [77], genetic algorithm [81], and mixed integer linear programming [77,78]. However, fast and effective reconfiguration response for electric service restoration is one of the critical requirements, which can be achieved by using state-of-the-art AI techniques to solve the reconfigurable power grids. Therefore in this section, a deep learning model is applied to solve the reconfigurable power grids.

In this section a new AI technique, known as the deep learning gated recurrent unit (GRU) (DLGRU), is developed to solve the reconfigurable power grids by learning the topological patterns of buses/lines with their physical features. GRU was first designed in Ref. [82] to decrease the complexity of long short-term memory (LSTM) and also enhance the LSTM performance. Same as LSTM, GRU has gates that can control the flow of the information from the input to the output. DLGRU can potentially understand complex nonlinear topological characteristics of the grid. Moreover, as will be shown in the result section, the total operation cost of the power grid by utilizing the DLGRU technique to select optimal switching of the reconfiguration is almost similar to conventional optimization techniques. DLGRU can solve the reconfigurable power grids in real time. However, conventional optimization techniques require

a certain amount of time interval to obtain the optimal results by running the optimal power flow. More information regarding the GRU method can be obtained in Ref. [82].

As mentioned, DLGRU is proposed to solve the reconfigurable power grid with the following objective function and constraints through a "learn to optimize" manner. Specifically, the main objective is to minimize the total operation cost as:

$$\min \sum_{i\in\Omega^{\mathrm{DG}}}\sum_{t\in\Omega^{T}}[C_i P_{it}^{G}+\mathrm{SU}_{it}+\mathrm{SD}_{it}]+\sum_{k\in\Omega^{S}} N_{\mathrm{RCS},kt}\lambda_{\mathrm{RCS}} \tag{7.19}$$

where $C_i$ is the generation cost of the $i$th unit, $P_{it}^{G}$ is the generated active power of the $i$th unit at time $t$, and $\mathrm{SU_{it}}$ and $\mathrm{SD_{it}}$ are the start-up and shutdown costs of the $i$th unit at time $t$, respectively. Also, $N_{\mathrm{RCS},kt}$ and $\lambda_{\mathrm{RCS}}$ represent the number of reconfigurable switching actions of $k$th remote-control switch (RCS) at time $t$ and reconfiguration switching cost, respectively. Moreover, $\Omega^{\mathrm{DG}}$, $\Omega^{T}$, and $\Omega^{S}$ denote sets of the generation units, switches, and time, respectively. It should be noted that the first and second terms of Eq. (7.19) represent the generation and reconfiguration switching costs of the grid, respectively. The proposed reconfigurable power grid includes some significant constraints in the following paragraphs.

Power balance constraints: the active and reactive power balances of each bus should be constrained as:

$$\sum_{nm\in\Omega^{L}}\left[P_{nm,t}^{L}-R_{nm}\left(I_{nm,t}^{L}\right)^{2}\right]-\sum_{nm\in\Omega^{L}}\left[P_{nm,t}^{L}\right]+P_{it}^{G}=P_{t}^{D}\forall i\in\Omega^{\mathrm{DG}},\forall t\in\Omega^{T} \tag{7.20}$$

$$\sum_{nm\in\Omega^{L}}\left[Q_{nm,t}^{L}-X_{nm}\left(I_{nm,t}^{L}\right)^{2}\right]-\sum_{nm\in\Omega^{L}}\left[Q_{nm,t}^{L}\right]+Q_{it}^{G}=Q_{t}^{D}\forall i\in\Omega^{\mathrm{DG}},\forall t\in\Omega^{T} \tag{7.21}$$

where $P_{nm,t}^{L}$ and $Q_{nm,t}^{L}$ are active and reactive power flow of line $nm$ at time $t$, respectively. $P_{t}^{D}$ and $Q_{t}^{D}$ are the total active and reactive power demand at time $t$, respectively. $R_{nm}$ and $X_{nm}$ are resistance and reactance of line $nm$, respectively. Also, $I_{nm,t}^{L}$ represents the current flow of distribution line $nm$ at time $t$. It should be noted that $\Omega^{L}$ denotes as the set of distribution lines. To apply the KVL to the distribution lines, the following constraints should be considered:

$$(V_{mt})^2 - (V_{nt})^2 = 2\left(R_{nm}P^L_{nm,t} + X_{nm}Q^L_{nm,t}\right) - (Z_{nm})^2\left(I^L_{nm,t}\right)^2 + \Delta V_{nmt}$$
$$\forall nm \in \Omega^L, \forall n, m \in \Omega^N, \forall t \in \Omega^T \tag{7.22}$$

$$(V_{mt})^2\left(I^L_{nm,t}\right)^2 = \left(P^L_{nm,t}\right)^2 + \left(Q^L_{nm,t}\right)^2 \forall nm \in \Omega^L, \forall n, m \in \Omega^N, \forall t \in \Omega^T \tag{7.23}$$

where $n$ and $m$ are defined as the bus indices, and $\Omega^N$ denotes the set of buses. Here, $V_{mt}$ represents the voltage of bus $m$ at time $t$, and $\Delta V_{nmt}$ is the auxiliary variable that can be zero if line $mn$ is switched on at time $t$. Otherwise, it can be positive or negative, which depends on the difference between the voltages of the sending and receiving ends of line $nm$. Finally, $Z_{nm}$ denotes the impedance of line $nm$.

Generation units constraints: the active and reactive powers of generation units are constrained to a minimum and maximum capacity as:

$$\underline{P^G_i} I_{it} \le P^G_{it} \le \overline{P^G_i} I_{it} \forall i \in \Omega^{DG}, \forall t \in \Omega^T \tag{7.24}$$

$$\underline{Q^G_i} I_{it} \le Q^G_{it} \le \overline{Q^G_i} I_{it} \forall i \in \Omega^{\mathrm{DG}}, \forall t \in \Omega^T \tag{7.25}$$

where $\underline{P^G_i}$ and $\overline{P^G_i}$ present the minimum and maximum active power capacity of the $i$th unit, respectively. Similarly, $\underline{Q^G_i}$ and $\overline{Q^G_i}$ present the minimum and maximum active power capacity of the $i$th unit, respectively. It should be noted that $I_{it}$ is the on/off status of the $i$th unit at time $t$, which can be zero (when unit is on) or one (when unit is off).

Bus voltage and angle limits: the voltage and angle of each bus are limited as:

$$\underline{V_n} \le V_{nt} \le \overline{V_n} \forall n \in \Omega^N, \forall t \in \Omega^T \tag{7.26}$$

$$-\pi \le \theta_{nt} \le \pi \forall n \in \Omega^N, \forall t \in \Omega^T \tag{7.27}$$

where $\underline{V_n}$ and $\overline{V_n}$ represent the minimum and maximum permissible voltage at bus $n$, respectively. Also, $\theta_{nt}$ denotes the angle of bus $n$ at time $t$.

Reconfiguration constraints: the number of reconfigurable switching per day is constrained as:

$$N_{\mathrm{RCS},k,t} \le \overline{N_{\mathrm{RCS}}} \forall k \in \Omega^S, \forall t \in \Omega^T \tag{7.28}$$

where $\overline{N_{RCS}}$ denotes the maximum permissible switching per day. Also, the radiality of the network after each reconfiguration process is assured:

$$N_{loop} = N_{branch} - N_{bus} + 1 \tag{7.29}$$

where $N_{loop}$, $N_{branch}$, and $N_{bus}$ represent the total number of network main loops, the number of branches, and the number of buses, respectively. More details regarding the reconfigurable power grid as well as the conventional methods for solving the problem can be found in Ref. [77].

Fig. 7.7 illustrates the proposed DLGRU to solve the reconfiguration of the power systems. There are inputs (the load and the generation unit powers), output (the switching numbers of the reconfigurations of the power systems), and hidden GRU layers.

### 7.3.3 Network reconfiguration results

In this part, as shown in Fig. 7.8, a modified IEEE 33-bus test system is selected to demonstrate the effectiveness of the proposed DLGRU for finding the optimal switching of the reconfigurable power grids. The model includes five tie switches (that are shown by the *dotted lines*), as well as 33 sectionalized switches (that are shown as the *solid lines*). Table 7.8 shows the characteristics of the generation units within the network. It is worth noting that in any time interval, five switches should be open to maintain the radiality of the network. The number of hidden layers in the simulation results is 4, and there are 50 units in each layer.

The reconfiguration switching for both DLGRU and a conventional benchmark method [77] for a 24-hour time horizon is compared in

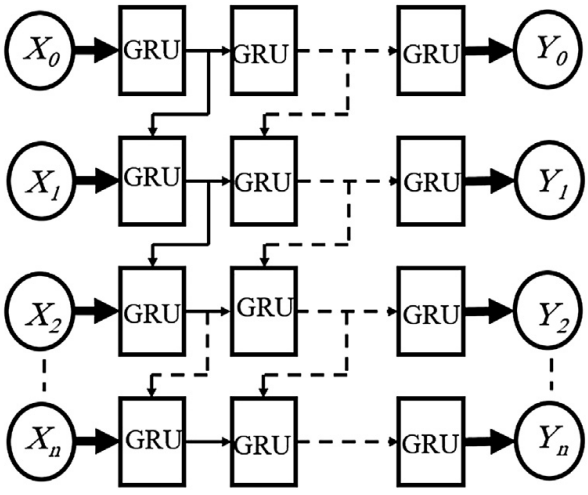


**Figure 7.7** DLGRU block diagram. *DLGRU*, Deep learning gated recurrent unit.

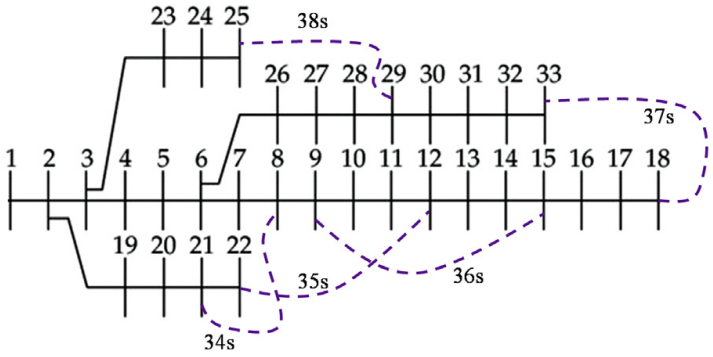


**Figure 7.8** Single line diagram of the modified IEEE 33-bus system.

**Table 7.8** Generation units features.

| Generation type | $\underline{P_i^G}$ (kW) | $\overline{P_i^G}$ (kW) | $C_i$ ($/kWh) | SU/SD ($) |
|---|---|---|---|---|
| WT1 | 0 | 25 | 1.073 | 0 |
| WT2 | 0 | 25 | 1.073 | 0 |
| FC | 80 | 1000 | 0.294 | 0.95 |
| MT1 | 100 | 1500 | 0.457 | 1.65 |
| MT2 | 100 | 1500 | 0.457 | 1.65 |

*FC*, Fuel cell; *MT1*, microturbine 1; *MT2*, microturbine 2; *WT1*, wind turbine 1; *WT2*, wind turbine 2.

Fig. 7.9. It is observed that there is a small difference between the optimal result of the proposed DLGRU method and the conventional technique for generating the switching's of the lines to reconfigure the network. Fig. 7.10 shows the contribution of the generation units for both DLGRU and the conventional technique. The total operation cost of the conventional method and the DLGRU method are $138,017.4 and $138,132.2, respectively. Based on the simulation results, the operation costs of the proposed machine learning technique and the conventional technique are very close. However, the proposed DLGRU technique can find the optimal switching in real time due to its fast convergence speed.

## 7.4 Advanced artificial intelligence and machine learning applications to building occupancy detection

The occupancy detection is beneficial to improve building energy management and provide important information for demand response.

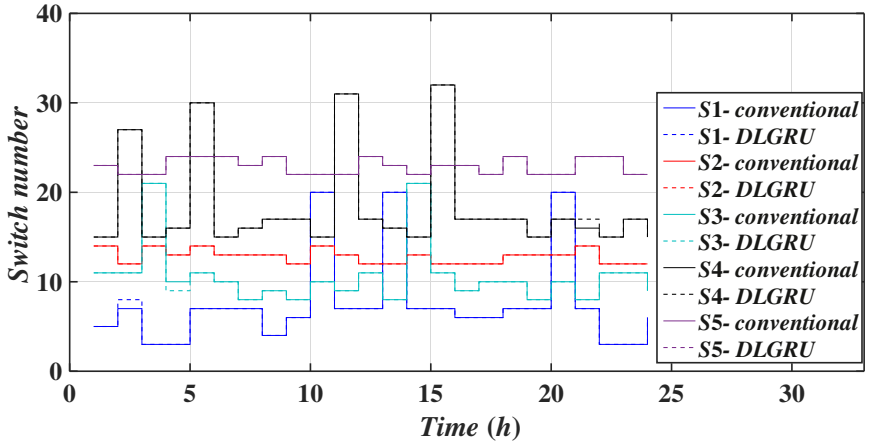


**Figure 7.9** Reconfiguration switching of both the conventional and DLGRU techniques. *DLGRU*, Deep learning gated recurrent unit.

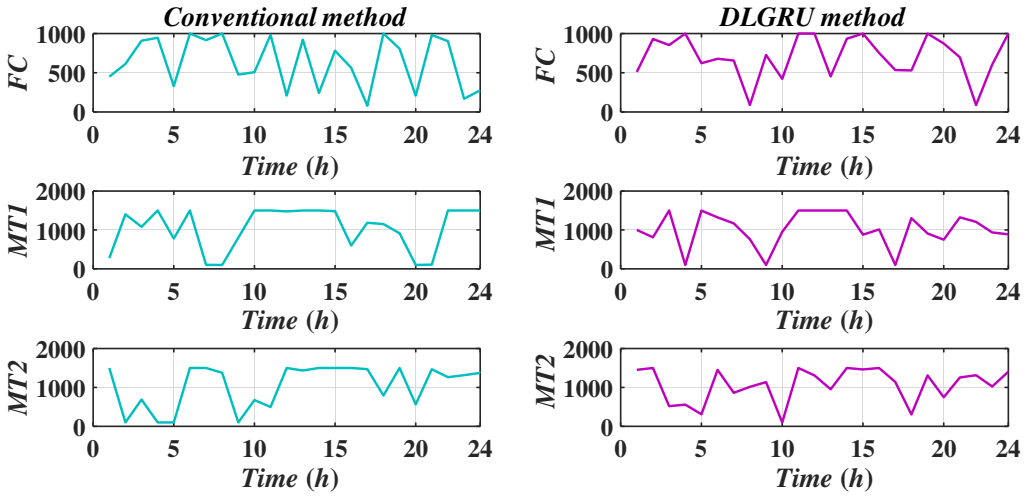


**Figure 7.10** Generation units' power. Left-hand side: conventional method; right-hand side: DLGRU method. *DLGRU*, Deep learning gated recurrent unit.

This section will first review the state-of-the-art of AI-based occupancy detection. Then, a novel deep learning–based method is developed and validated by publicly available dataset.

## 7.4.1 Prior research work

Based on the monitored objects, occupancy detection can be grouped into intrusive and nonintrusive approaches. Intrusive sensors directly measure indoor environments, including motional, acoustic, or climatic parameters [83]. For example, Candanedo and Feldheim [84] compared occupancy detection methods with different indoor climate feature combinations, among which the best model had over 99% accuracy.

Surveillance videos were used for occupancy detection with 95.3% accuracy in Ref. [85]. However, the intrusive occupancy detection is challenging to extensively deploy due to the high installation cost, additional operation requirements (e.g., the illumination condition for cameras), and privacy concerns. Therefore it motivated the development of nonintrusive occupancy detection. The nonintrusive occupancy detection relies on infrastructure sensors that monitor parameters such as Wi-Fi, Bluetooth, or radio-frequency identification (RFID) [86–88]. For example, Wang et al. [89] developed a Wi-Fi–based occupancy detection system with 72.7% accuracy, which helped save 26.4% of energy in cooling and ventilation demands. An RFID-based occupancy detection system was able to track the stationary and mobile occupants with 88% and 62% accuracy, respectively [87]. The nonintrusive methods have fewer privacy issues; however, they still suffer from possibly unsatisfactory accuracies, low infrastructure/device coverage, and extra occupant participation. The limitations of both intrusive and nonintrusive sensors provide an opportunity to infer building occupancy from AMI data. Nevertheless, AMI-based occupancy detection is still limited compared to other sensor-based detection.

Regarding the detection algorithms, data-driven machine learning algorithms are widely applied, due to the growing penetration of sensors. For example, a collection of machine learning models, including a decision tree (DT) model, a SVM, and a Bayes network, were compared in Ref. [90], among which DT was the best model. In addition, SVM beat the hidden Markov model (HMM) and the *k*NN model with 80% accuracy [91]. Moreover, Jin et al. [92] developed an RF model, which outperformed HMM and SVM models with around 90% accuracy. In contrast with the wide deployment of shallow machine learning algorithms, deep learning is far from fully explored in occupancy detection problems. A CNN is the first deep learning model developed for occupancy detection, which provided occupancy information from indoor climate measurements with 95.42% accuracy [93]. Another deep learning architecture utilized an autoencoder long-term recurrent convolutional network, which identified the occupant activity from Wi-Fi–enabled Internet of Things devices [94].

To bridge the gap in occupancy detection using AMI data, a deep learning architecture is proposed in this section. The developed deep learning architecture stacks a CNN and a LSTM network sequentially. The developed CNN–LSTM architecture is expected to capture both

spatial and contextual representations of the AMI data. The problem formulation and methodology description are introduced in the rest of this section. The developed model is then compared with state-of-the-art classifiers with a publicly available dataset.

## 7.4.2 The convolutional neural network–long short-term memory deep learning architecture

### 7.4.2.1 Occupancy detection problem formulation

The objective of the building occupancy detection is to identify the real-time occupancy condition, $\mathbf{y} \in R^{N \times 1}$, of a house from its AMI data, $\mathbf{X} \in R^{N \times F}$, by using a stacking CNN model and an LSTM model [95]:

$$\hat{\mathbf{y}} = F(\mathbf{X}, \mathbf{W}) = F_R\left\{F_C(\mathbf{X}, \mathbf{W}_C), \mathbf{W}_R\right\} \tag{7.30}$$

where $N$ and $F$ are the sample size and dimension, respectively. $\mathbf{y}$ and $\hat{\mathbf{y}}$ are actual and detected occupancy conditions, respectively. $F(\cdot)$, $F_C(\cdot)$, and $F_R(\cdot)$ indicate the developed CNN–LSTM model and its CNN and LSTM components, respectively. $\mathbf{W}$, $\mathbf{W_C}$, and $\mathbf{W_R}$ are the parameters in the according models. With the condition of either occupied or vacant at every timestamp, that is, $y \in 0, 1$, the occupancy detection is a binary classification problem. Therefore the objective is to minimize the loss function, which is based on the weighted binary cross-entropy, given by:

$$J(\mathbf{W}) = -\frac{1}{N}\sum_{n=1}^{N}\left[\omega y_n \log\left(\hat{y}_n\right) + (1-\omega)(1-y_n)\log\left(1-\hat{y}_n\right)\right] \tag{7.31}$$

where $\omega$ is the binary cross-entropy weight, defined as $\omega = P_Y[y = 0 | y \in \mathbf{y}]$.

### 7.4.2.2 Convolutional neural network

With one or multiple feature learning blocks (FLBs) that consist of a convolutional layer and a max-pooling layer, CNN has powerful feature learning ability. A convolutional layer (indexed by $l$) contains $D^{(l+1)}$ filters. A convolution operation is performed in each filter to construct a set of feature maps as:

$$\mathbf{Z}^l = \mathbf{W}^l \times \mathbf{X}^l + \mathbf{b}^l \tag{7.32}$$

where $\mathbf{X}^l \in R^{H^l \times D^l}$ is the convolutional layer input. $\mathbf{Z} \in R^{H^{(l+1)} \times D^{(l+1)}}$, $\mathbf{W} \in R^{H^{(l+1)} \times D^l \times D^{(l+1)}}$, and $\mathbf{b} \in R^{H^{(l+1)} \times D^{(l+1)}}$ are the feature map matrix,

layer parameter tensor, and bias matrix, respectively. Eq. (7.32) can be expressed in detail as:

$$z^{l}_{h^{(l+1)},d^{(l+1)}} = \sum_{h^l=1}^{H^l}\sum_{d^l=1}^{D^l}\left[w^{l}_{d^l,h^l,d^{(l+1)}} \times x^{l}_{h^l,d^l} + b^{l}_{h^{(l+1)},d^{(l+1)}}\right] \tag{7.33}$$

where $(h, d)$ is a doublet index used to locate the element, $x$, in $\mathbf{X}$.

To add nonlinearity to the network an element-wise rectified linear unite (ReLU) is used to activate convolution outputs. The ReLU activation function is selected due to its computational efficiency, better convergence, superior performance, and amelioration of vanishing gradients compared to other functions [96]. The ReLU function is expressed as:

$$z^{l}_{h^l,d^l} = \max(0, x^{l}_{h^l,d^l}) \tag{7.34}$$

A max-pooling layer is introduced as the last layer of an FLB to achieve more translation invariance during the spatial representation learning. In this study a unified nonoverlapping moving window is used to subsample the activated convoluted feature maps by a factor of 2. The max-pooling layer is expressed as:

$$z^{l}_{h^{(l+1)},d^l} = \max\left\{\mathbf{x}^l | \mathbf{x}^l : = x^{l}_{h^l:h^l+\left(H^l/(H^{l+1})\right),d^l}\right\} \tag{7.35}$$

#### 7.4.2.3 Long short-term memory network

RNNs have been effectively applied in time series data analytics due to its capability of capturing temporal correlations. The LSTM is a variant of RNN, which avoids the vanishing gradient problem by gated regulators. A typical LSTM block contains a memory cell, a forget gate, an input gate, and an output gate, shown as Fig. 7.11. The three gates have different functions, where the input gate determines the new information stored in the memory, the forget gate decides the useless old information to exclude, and the output gate exploits useful information to output from the memory cell. The tensor operations in the gates of a forward LSTM are expressed as:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_f) \tag{7.36}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_i) \tag{7.37}$$

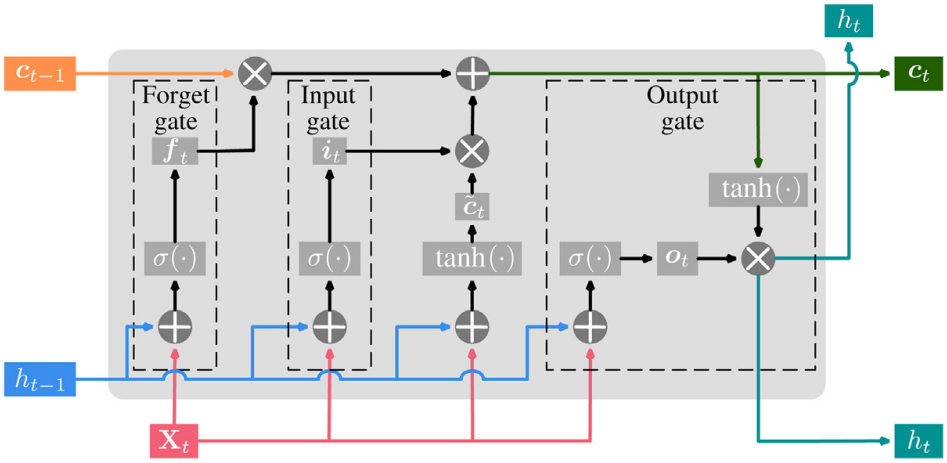


**Figure 7.11** The LSTM building block. *LSTM*, Long short-term memory.

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_c) \tag{7.38}$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{t-1} + \mathbf{i}_t \times \tilde{\mathbf{c}}_t \tag{7.39}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_o) \tag{7.40}$$

$$\mathbf{h}_t = \mathbf{o}_t \times \tanh(\mathbf{c}_t) \tag{7.41}$$

where $\mathbf{f}$, $\mathbf{i}$, $\mathbf{o}$, and $\mathbf{c}$ are the activation vectors of the forget, input, output gates and the memory cell, respectively. $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_c$, and $\mathbf{W}_o$ are the input matrices of gates or memory cell. $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_c$, and $\mathbf{b}_o$ are the corresponding bias vectors. $\sigma(\cdot)$ and $\tanh(\cdot)$ are the sigmoid and hyperbolic tangent activation functions, respectively. $\mathbf{h}$ is the hidden state and the output of an LSTM hidden layer. $\tilde{\mathbf{c}}$ is the new state candidate vector. The bracket is the concatenation operator. The nodal connections in an LSTM cell are shown in Fig. 7.11.

One typical issue of most recurrent models is that they can only deal with unidirectional dependencies in the data. To capture both forward and backward dependencies in the AMI data a bidirectional LSTM (BiLSTM) layer is included in the LSTM configuration. The nodal connection and the tensor calculations in a BiLSTM are almost the same with those in a unidirectional LSTM, except for the processing directions. The bidirectional operations within a BiLSTM can be expressed as:

$$\vec{\mathbf{f}}_t = \sigma\left(\vec{\mathbf{W}}_f\left[\vec{\mathbf{h}}_{t-1}, \vec{\mathbf{X}}_t\right] + \vec{\mathbf{b}}_f\right) \tag{7.42}$$

$$\overleftarrow{\mathbf{f}}_t = \sigma\left(\overleftarrow{\mathbf{W}}_f\left[\overleftarrow{\mathbf{h}}_{t+1}, \overleftarrow{\mathbf{X}}_t\right] + \overleftarrow{\mathbf{b}}_f\right) \tag{7.43}$$

where $\vec{}$ and $\overleftarrow{}$ denote the forward and backward operations, respectively. Both $\vec{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are generated and concatenated as the output of the BiLSTM:

$$\mathbf{h}_t = [\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t] \tag{7.44}$$

#### 7.4.2.4 The developed convolutional neural network–long short-term memory architecture

The overall framework of the developed CNN–LSTM and the tensor manipulations are shown in Fig. 7.12. The developed framework contains three stacking components, which are a CNN network, an LSTM, and a dense layer configuration. The AMI data is first convoluted through the CNN network to generate spatial feature maps, as shown in the bottom left dashed box. Inspired by the VGGNet, two FLBs with four layers (i.e., two convolutional layers and two max-pooling layers) are consisted in the CNN. There are 128 and 64 filters in the two FLBs, respectively, which are used to extract the high-level abstract spatial features from the AMI data.

The output of CNN configuration serves as the input to LSTM configuration. To better capture the contextual patterns in the AMI data, we increase the depth of the architecture by stacking three LSTM layers vertically [97]. As shown in Fig. 7.12, the first LSTM layer extracts temporal features from the previous CNN output and generates hidden states. Then, a BiLSTM layer takes both forward and backward dependencies into account, which is combined by the last LSTM layer. The developed BiLSTM configuration is expected to learn hierarchical representations of the convolutional time series of AMI data by operating hidden states at different timescales.

The last component in the CNN–LSTM model is a dense layer configuration with two fully connected layers and a final classification layer. The output of LSTM is first flattened and fed into the first dense layer, whose output is then fed into the second dense layer:

$$\mathbf{Z}^l = \mathbf{W}^l \cdot \mathbf{X}^l + \mathbf{b}^l \tag{7.45}$$

where all the inputs are transmitted to the output. The last layer is a classification layer with sigmoid activation function due to the mutual-exclusive character of occupancy detection results:

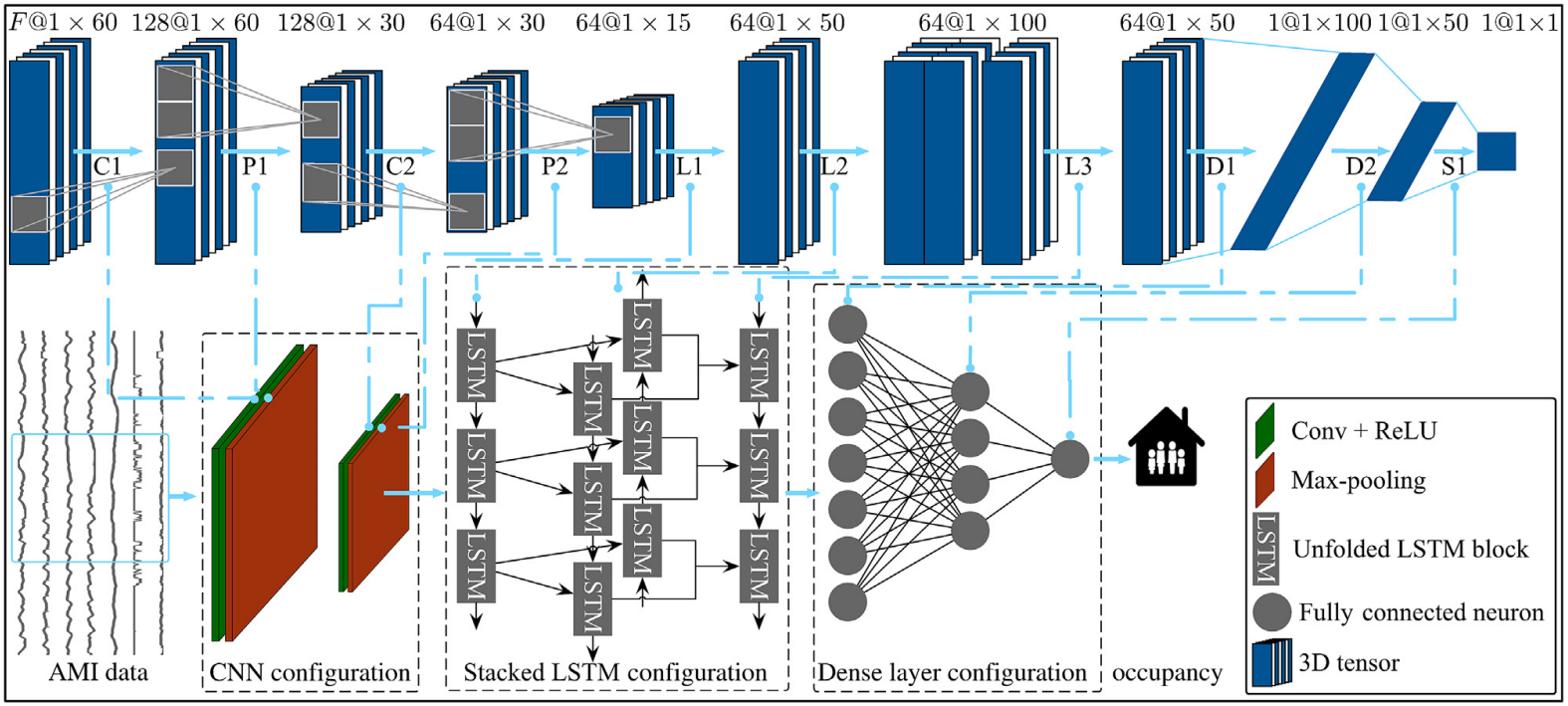


**Figure 7.12** The developed end-to-end CNN−LSTM occupancy detection framework. *CNN*, Convolutional neural network; LSTM, long short-term memory.

$$\mathbf{Z}^l = \frac{1}{1 + e^{-\mathbf{W}^l \cdot \mathbf{X}^l + \mathbf{b}^l}} \tag{7.46}$$

where $l = 10$. The occupancy condition could be determined by a applying a threshold to the last-layer output:

$$\hat{y} = H(z - th) \tag{7.47}$$

where $H(\cdot)$ is the Heaviside step function and $th = 0.5$ is the threshold value.

### 7.4.3 Experiments

The developed CNN−LSTM model is tested with the largest publicly available dataset with both AMI and occupancy data, that is, the Electricity Consumption and Occupancy dataset [98]. The dataset has summer and winter data of five houses with 1-second resolution (aggregated from data with 1 Hz frequency). To avoid the noise and redundancy, basic features were extracted from the AMI data [99]. The minute data is flattened every hour to fit the CNN input format with $\boldsymbol{W} = 60$. The quality-control with two criteria was applied to the dataset: (1) the data length should be more than 900 and (2) both occupancy labels (i.e., occupied and vacant conditions) should be more than 10% of the total occupancy data. As a result, four periods of data from three houses were qualified and selected for case studies, which are summarized in Table 7.9. The ratio of training, validation, and testing data is 3:1:1 for each case study.

Hyperparameters of the developed CNN−LSTM model are determined by a trial-and-error manner with the training and validation dataset and are listed in Table 7.9. The model has a total of 240,489 trainable parameters (2.5 MB), which is a relatively small network, compared to other famous deep learning architectures (e.g., the 16-layer VGGNet has 528 MB weights). Before being trained, parameters in each layer are

**Table 7.9** Data summary of case studies.

| Case notations | Data dimension ($N$, $W$, $F$) | Label (occupied/vacant) |
|---|---|---|
| C1 | (935, 60, 30) | 774/161 |
| C2 | (1103, 60, 30) | 831/272 |
| C3 | (1079, 60, 24) | 771/308 |
| C4 | (1367, 60, 29) | 1044/323 |

*Note*: The case notations (C1−C4) are different from Sections 7.2.3.1 and 7.2.3.2.

**Table 7.10** Hyperparameters of layers in the developed convolutional neural network–long short-term memory (LSTM) model.

| Layer | Type | Hyperparameter | Trainable parameters |
|---|---|---|---|
| C1 | Convolutional | Input size: $60 \times F$<br>Filter size: 3<br>Filter number: 128 | 11,648 |
| P1 | Pooling | Window size: $2 \times 128$<br>Stride: 2 | 0 |
| C2 | Convolutional | Input size: $30 \times 128$<br>Filter size: 3<br>Filter number: 64 | 34,640 |
| P2 | Pooling | Window size: $30 \times 64$<br>Stride: 2 | 0 |
| L1 | LSTM | Output length: 50 | 23,000 |
| L2 | BiLSTM | Output length: 200 | 120,800 |
| L3 | LSTM | Output length: 50 | 50,200 |
| D1 | Dense | Neurons: 100 | 5100 |
| D2 | Dense | Neurons: 50 | 5050 |
| S2 | Classification | Neurons: 1 | 51 |

initialized with the Xavier method, where biases are initialized as zeros and weights conform the Gaussian distribution [100]. Twenty percent of neurons in the convolutional, LSTM, and dense layers are randomly dropped to overcome the possible overfitting [101] (Table 7.10).

In this study, mini-batch stochastic gradient descent (SDG) is selected as the optimizer to train the developed CNN–LSTM model. The SDG minimizes the objective function $\boldsymbol{J(W)}$ in Eq. (7.31) by updating the parameters in the opposite direction of its gradients. The complete data is passed forward and backward through the network with 100 epochs. The mini-batches ($\boldsymbol{B := [X,y]}$), with a batch size of 30, are randomly generated to shuffle the data order in each epoch. Gradients in each iteration are calculated by averaging over the mini-batch. The learning rate scheduling is used to dampen training oscillations. Specifically, the training starts with a learning rate of 0.1 and reduces the learning rate by 50% when a plateau is reached for more than 10 epochs. To reduce the risk of convergence to local minima, momentum is adopted in the training. The weights are updated with the above techniques as:

$$\mathbf{V}_{i+1} = \gamma \mathbf{V}_{\mathrm{i}} + \eta_{i+1} \nabla_{\mathbf{W}} \mathbf{J}(\mathbf{W};\mathbf{B}) \tag{7.48}$$

$$\mathbf{W}_{i+1} = \mathbf{W}_i - \mathbf{V}_{i+1} \tag{7.49}$$

where $\boldsymbol{i}$ is the iteration index, $\boldsymbol{V}$ is the weight update matrix, $\boldsymbol{\gamma = 0.9}$ is the momentum, and $\boldsymbol{\eta}$ is the learning rate.

Six state-of-the-art machine learning–based classifiers are adopted to compare with the developed CNN–LSTM method, which are a *k*NN model, an SVM model with linear kernel, a GP model, an RF, a multilayer perceptron (MLP) classifier, and an adaptive boosting model (AdaBoost). We believe the selected benchmarks are general and representative, since they cover the nonparametric model (*k*NN), kernel-based model (SVM), feedforward NN (MLP), and ensemble learning models (RF and AdaBoost). The hyperparameters and parameters of these models are determined empirically and listed in Table 7.11, including the number of neighbors (`n_neighbors`) in *k*NN, the penalty weight (C) in SVM, the covariance function (`kernel`) of the GP, the number of trees (`n_estimators`), the maximum depth of the tree (`max_depth`), the number of features for the best split (`max_features`) in RF, the number of hidden layer neurons (`neuron`), activation function (`activation`), regularization penalty (`alpha`) in MLP, and the maximum number of estimators (`n_estimators`), and learning rate (`learning_rate`) in AdaBoost.

### 7.4.4 Results

The case studies are implemented in Python version 3.6 with the `Keras`, `Tensorflow`, and `scikit-learn` libraries. The experiments are repeated 10 times to improve the reproducibility and consistency. All the experiments are conducted on a workstation with an Intel Xeon(R) E5-2603 1.6 GHz CPU and an NVIDIA TITAN V GPU. It took 11 minutes to train a CNN–LSTM model with the experiment setups described above and 3.7 ms to detect the occupancy at a time step. The computational time is applicable for real-time detection.

**Table 7.11** Benchmark model hyperparameters and parameters.

| Model | Hyperparameters/parameters |
|---|---|
| ***k*NN** | `n_neighbors` = **3** |
| SVM | `C` = 0.025 |
| GP | `kernel` = $\boldsymbol{1.0 \times RBF(1.0)}$ |
| RF | `n_estimators` = 10, `max_depth` = 5, `max_features` = 1 |
| MLP | `neuron` = 100, `activation` = ReLU, `alpha` = 0.01 |
| AdaBoost | `n_estimators` = 50, `learning_rate` = 1.0 |

*RF*, random forest; *GP*, Gaussian process; *kNN*, *k*-nearest neighbor; *SVM*, support vector machine; *MLP*, multilayer perceptron classifier.

Being a typical binary classification problem, the occupancy detection can be evaluated based on the binary confusion matrix, which consists of four elements (as shown in Fig. 7.13): (1) true positive (TP), denoting the count that a house is actually occupied and is detected occupied; (2) false positive (FP), denoting the count that a house is actually vacant but is detected occupied; (3) true negative (TN), denoting the count that a house is actually vacant and is detected vacant; and (4) false negative, denoting the count that a house is actually occupied but is detected vacant. Based on the confusion matrix, five metrics are used to quantify the overall performance of the classification models, which are accuracy (ACC), sensitivity (SNS, also known as TP rate or recall), specificity (SPC, also known as TN rate), precision (PRC), and $F1$ score ($F1$):

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{7.50}$$

$$\mathrm{SNS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{7.51}$$

$$\mathrm{SPC} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{7.52}$$

$$\mathrm{PRC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{7.53}$$

$$F1 = 2\frac{\mathrm{SNS} \times \mathrm{PRC}}{\mathrm{SNS} + \mathrm{PRC}} \tag{7.54}$$

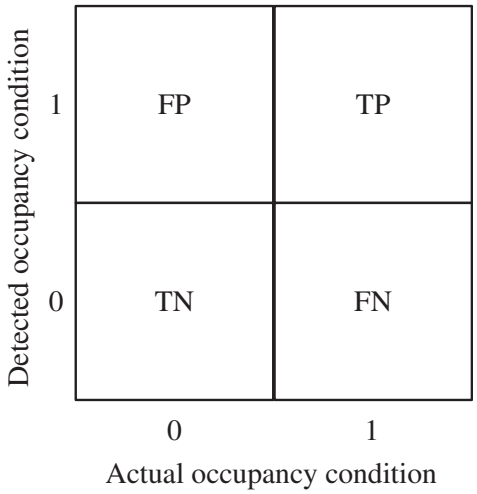


**Figure 7.13** A binary confusion matrix.

where ACC indicates the overall accuracy of the detection. SNS measures the proportion of the actual occupied conditions that are correctly detected as such. SPC measures the proportion of the actual vacant conditions that are correctly detected as such. PRC is the success probability of detecting a correct occupied condition. $F1$ combines several metrics in the imbalanced classification problem.

The occupancy detection accuracy is not only affected by the models, but also related to the threshold shown in Eq. (7.47). Therefore in addition to the five overall metrics, another set of metrics are used to assess the goodness of the classifiers over the entire operating range, which are receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC). The former one is a curve of TP rate (SNS) against FP rate (1 − SPC) at various threshold settings, which can also be used to determine the best *th*. A larger deviation between the ROC curve and the diagonal line represents a better occupancy detection. The AUC value is a comprehensive measurement of the ROC curve, where a larger AUC value (maximum AUC = 1) indicates a better result.

The confusion matrix of one repeat is shown in Fig. 7.14. Results of different models are located in different columns and matrices in different rows indicate results of different cases. Even though the labels are highly imbalanced in the whole dataset (as listed in Table 7.9), the testing data labels could be balanced. For example, the vacant conditions and occupied conditions ratio is 115:106 in the testing data of C. In addition, the diversity of the four cases directly impacts the model performance. For instance, *k*NN detects more FP than TN in C1 but vice versa in C2 and C4. Different models perform distinctively in the same case. For example, SVM, GP, and MLP accurately detect more occupied conditions; however, AdaBoost is more powerful in detecting the vacant conditions in C2. We can conclude that both the dataset and the benchmarks are diverse and general for the comparisons.

Based on confusion matrices, the evaluation metrics are calculated and listed in Tables 7.12 and 7.13. From Table 7.12, it is observed that every benchmark model has the chance to generate satisfactory occupancy detection, since five out of six benchmark models outperform others in some cases and based on some metrics. Nevertheless, none of the benchmark models can always beat others in all the four cases and all the metrics. In addition, some benchmark models make extremely assertive detections. For example, GP detects occupancy conditions with 100% SNS but 0% SPC in C1, C3, and C4. Some benchmarks are not robust,

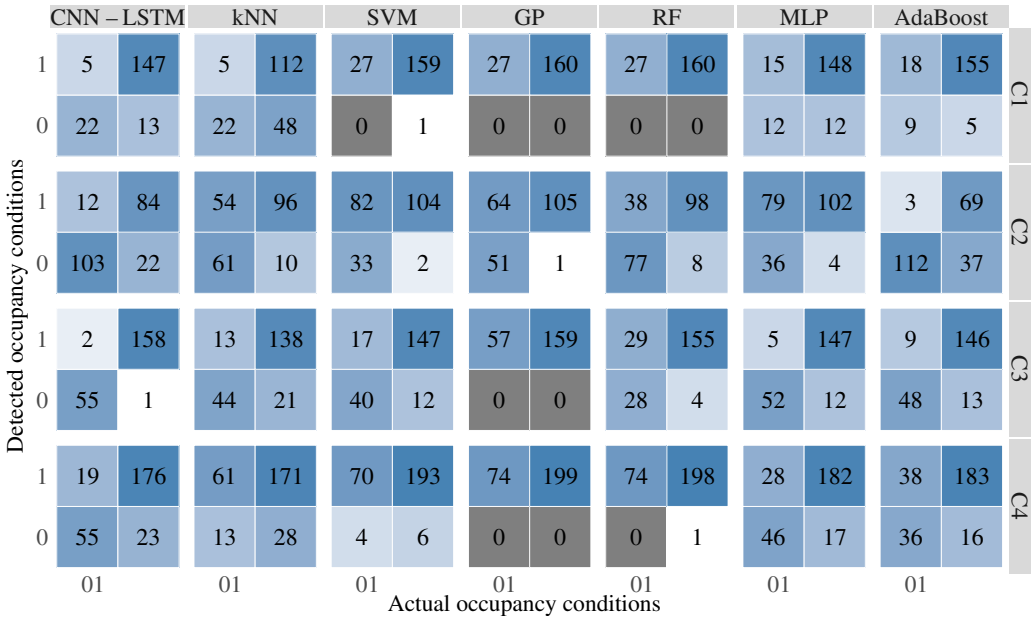


**Figure 7.14** Confusion matrices of the occupancy detection results. The darker color indicates a higher frequency, and the gray color means a zero occurrence. Positive diagonal elements indicate right detections, and negative diagonal elements show the wrong detections.

which is revealed by the nonnegligible variance of the metrics over 10 experiment repeats, such as SPC of RF in C2 and SPC of MLP in C4. In contrast, the developed CNN−LSTM model shows encouraging accuracy in both the occupied and vacant conditions. More importantly, the CNN−LSTM model is more accurate than benchmark models in all the four cases, indicated by two overall evaluation metrics, ACC and $F1$. The average ACC and $F1$ over the four cases and 10 experiment repeats are 89.41% and 91.55%, respectively. The robustness of the CNN−LSTM model is not only revealed in diverse cases but also shown in 10 sets of experiments, which is supported by the relatively small variances listed in Table 7.13.

To assess the model performance independent of the choice of *th*, ROC and AUC are adopted. Specifically, a set of *th* values, ranging from 0 to 1, are used to determine $\hat{\boldsymbol{y}}$ and calculate TPR and FPR, as shown in Fig. 7.15. The perfect classifier should have the ROC curve straight up the vertical axis then along the horizontal axis. The classifier that randomly generates occupancy detection results sits on the diagonal, and the classifier that detects complete reverse results has a curve in the bottom left part of the ROC space. Therefore the developed CNN−LSTM model has better and more robust occupancy detection capability with

**Table 7.12** Mean of evaluation metrics (%) of the 10 occupancy detection experiment repeats.

| Case | Metric | Models | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **CNN−LSTM** | ***k*NN** | **SVM** | **GP** | **RF** | **NN** | **AdaBoost** |
| C1 | ACC | 91.34 | 71.66 | 85.03 | 85.56 | 85.51 | 85.29 | 87.70 |
| | SNS | 93.13 | 70.00 | 99.38 | 100.00 | 99.94 | 91.06 | 96.88 |
| | SPC | 80.74 | 81.48 | 0.00 | 0.00 | 0.00 | 51.11 | 33.33 |
| | PRC | 96.63 | 95.73 | 85.48 | 85.56 | 85.55 | 91.74 | 89.60 |
| | *F*1 | 94.84 | 80.87 | 91.91 | 92.22 | 92.19 | 91.37 | 93.09 |
| C2 | ACC | 85.25 | 71.04 | 61.99 | 70.59 | 80.81 | 65.43 | 81.90 |
| | SNS | 82.36 | 90.57 | 98.11 | 99.06 | 89.62 | 94.43 | 65.09 |
| | SPC | 87.91 | 53.04 | 28.70 | 44.35 | 72.70 | 38.70 | 97.39 |
| | PRC | 86.79 | 64.00 | 55.91 | 62.13 | 76.69 | 59.08 | 95.83 |
| | *F*1 | 84.21 | 75.00 | 71.23 | 76.36 | 82.12 | 72.48 | 77.53 |
| C3 | ACC | 98.47 | 84.26 | 86.57 | 73.61 | 84.26 | 91.76 | 90.74 |
| | SNS | 99.12 | 86.79 | 92.45 | 100.00 | 95.66 | 92.45 | 92.58 |
| | SPC | 96.67 | 77.19 | 70.18 | 0.00 | 52.46 | 89.82 | 85.61 |
| | PRC | 98.81 | 91.39 | 89.63 | 73.61 | 85.06 | 96.23 | 94.72 |
| | *F*1 | 98.96 | 89.03 | 91.02 | 84.80 | 89.99 | 94.29 | 93.64 |
| C4 | ACC | 82.56 | 67.40 | 72.16 | 72.89 | 73.15 | 80.04 | 80.22 |
| | SNS | 89.50 | 85.93 | 96.98 | 100.00 | 99.70 | 95.33 | 91.96 |
| | SPC | 63.92 | 17.57 | 5.41 | 0.00 | 1.76 | 38.92 | 48.65 |
| | PRC | 87.21 | 73.71 | 73.38 | 72.89 | 73.19 | 81.02 | 82.81 |
| | *F*1 | 88.20 | 79.35 | 83.55 | 84.32 | 84.41 | 87.48 | 87.14 |

*ACC*, Accuracy; CNN, convolutional neural network; *GP*, Gaussian process; *kNN*, *k*-nearest neighbor; *LSTM*, long short-term memory; *PRC*, precision; *RF*, random forest; *SNS*, sensitivity; *SPC*, specificity; *SVM*, support vector machine.

different thresholds. The AUC values of all the experiments are shown in Fig. 7.16, where the developed CNN−LSTM outperforms other models consistently.

To dig into the reason of better occupancy detection performance of the developed CNN−LSTM model, the detection probabilities and results of the developed model are compared to those of the three best benchmarks, that is, SVM, MLP, and AdaBoost. The outperformance of the CNN−LSTM model is twofold. First, the developed model is more accurate, revealed by the smaller deviations between detections and the actual conditions. Actually, the CNN−LSTM model only misclassifies the occupancy three times at three discrete hours, which is less harmful to the demand response decisions. In contrast, all the three best competing models generate more and longer period wrong detections. The second advantage is that the CNN−LSTM model is more confident in the detection it

**Table 7.13** Standard deviation evaluation metrics (%) of the 10 occupancy detection experiment repeats.

| Case | Metric | CNN−LSTM | *k*NN | SVM | GP | RF | NN | AdaBoost |
|---|---|---|---|---|---|---|---|---|
| C1 | ACC | 0.49 | 0.00 | 0.00 | 0.00 | 0.17 | 0.85 | 0.00 |
| | SNS | 0.59 | 0.00 | 0.00 | 0.00 | 0.20 | 2.52 | 0.00 |
| | SPC | 3.40 | 0.00 | 0.00 | 0.00 | 0.00 | 9.85 | 0.00 |
| | PRC | 0.56 | 0.00 | 0.00 | 0.00 | 0.02 | 1.34 | 0.00 |
| | *F*1 | 0.30 | 0.00 | 0.00 | 0.00 | 0.10 | 0.66 | 0.00 |
| C2 | ACC | 1.09 | 0.00 | 0.00 | 0.00 | 6.80 | 4.50 | 0.00 |
| | SNS | 6.55 | 0.00 | 0.00 | 0.00 | 4.15 | 4.41 | 0.00 |
| | SPC | 6.18 | 0.00 | 0.00 | 0.00 | 15.85 | 12.50 | 0.00 |
| | PRC | 4.40 | 0.00 | 0.00 | 0.00 | 9.92 | 4.11 | 0.00 |
| | *F*1 | 1.51 | 0.00 | 0.00 | 0.00 | 4.67 | 1.81 | 0.00 |
| C3 | ACC | 0.38 | 0.00 | 0.00 | 0.00 | 4.04 | 0.57 | 1.95 |
| | SNS | 0.44 | 0.00 | 0.00 | 0.00 | 1.98 | 0.89 | 1.59 |
| | SPC | 1.29 | 0.00 | 0.00 | 0.00 | 15.24 | 3.86 | 2.96 |
| | PRC | 0.45 | 0.00 | 0.00 | 0.00 | 3.98 | 1.33 | 1.11 |
| | *F*1 | 0.26 | 0.00 | 0.00 | 0.00 | 2.32 | 0.35 | 1.35 |
| C4 | ACC | 1.58 | 0.00 | 0.00 | 0.00 | 0.49 | 2.56 | 0.00 |
| | SNS | 3.95 | 0.00 | 0.00 | 0.00 | 0.35 | 3.03 | 0.00 |
| | SPC | 11.82 | 0.00 | 0.00 | 0.00 | 2.30 | 16.68 | 0.00 |
| | PRC | 3.66 | 0.00 | 0.00 | 0.00 | 0.42 | 3.86 | 0.00 |
| | *F*1 | 1.07 | 0.00 | 0.00 | 0.00 | 0.23 | 1.15 | 0.00 |

*ACC*, Accuracy; *CNN*, convolutional neural network; *GP*, Gaussian process; *kNN*, *k*-nearest neighbor; *LSTM*, long short-term memory; *PRC*, precision; *RF*, random forest; *SNS*, sensitivity; *SPC*, specificity; *SVM*, support vector machine.

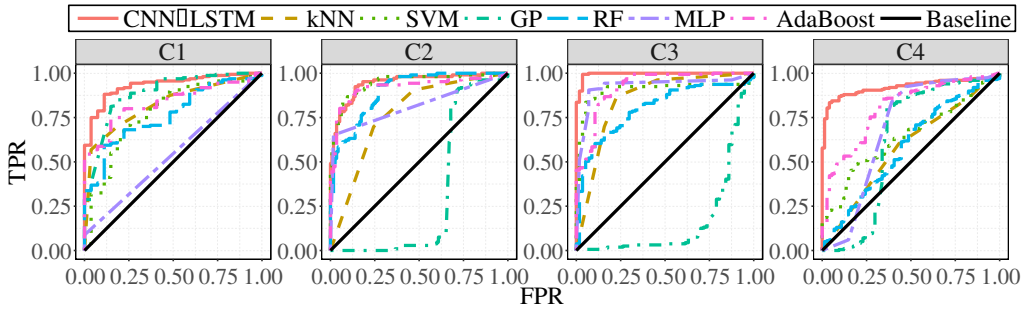


**Figure 7.15** ROC curves in randomly selected experiments. *ROC*, Receiver operating characteristic.

makes. This is illustrated by the darker color of the classification probabilities, compared to other models, such as AdaBoost. Therefore the outperformance of the developed model is well-explained (Fig. 7.17).

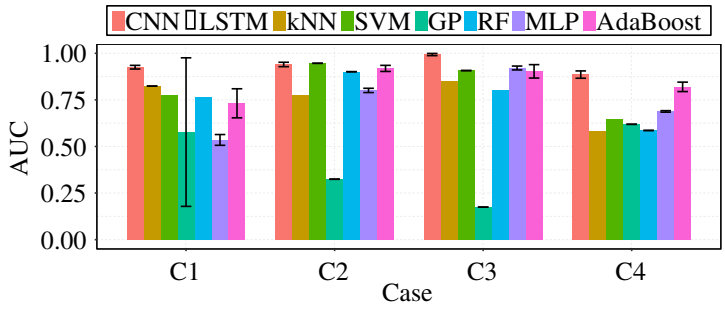


**Figure 7.16** The AUC statistics of the 10 experiment repeats. *AUC*, Area under ROC curve.

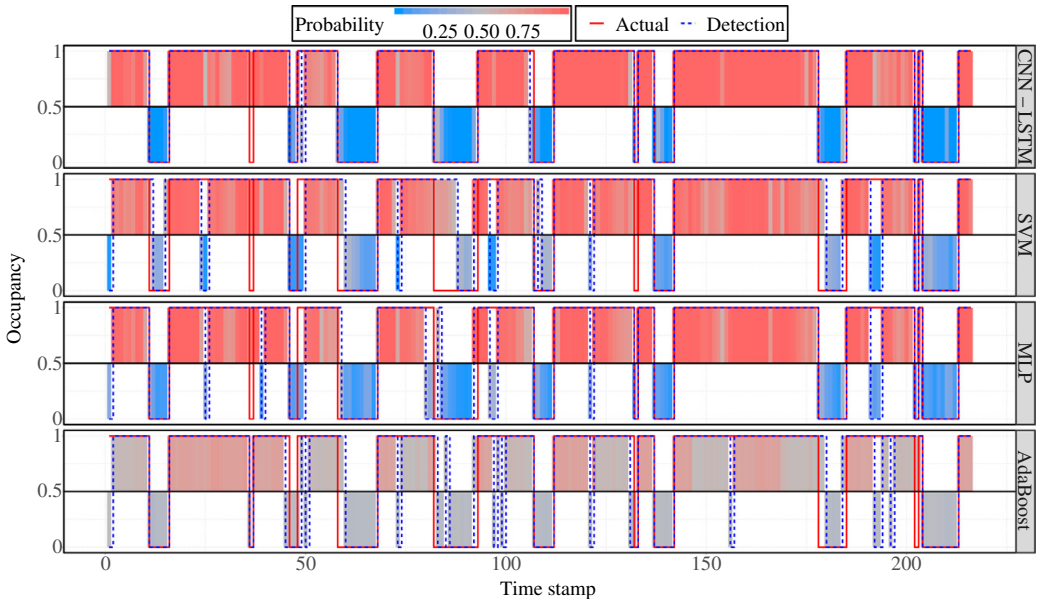


**Figure 7.17** The occupancy labels, detection probability, and results of C3.

## 7.5 Conclusion

AI has been witnessed to have significant impacts on power system operations. Large amounts of data are being collected by smart grid devices, such as the AMIs and the PMUs, which facilitate the wide applications of AI techniques to power systems. In this section, two emerging machine learning subfields were introduced: ensemble learning and deep learning. Three use cases were used to demonstrate their applications in power systems. Specifically, competitive and cooperative ensemble learning models were developed to provide short-term wind forecasts. Both methods included state-of-the-art machine learning models, for example, ANNs, SVR models, GBMs, RF models, and Q-learning models. In

addition, a DLGRU was used to solve a network reconfiguration problem through a "learn to optimize" manner. In the last application case, a deep learning configuration with a CNN and a LSTM network was developed to detect real-time occupancy conditions in smart buildings. Three sets of case studies based on publicly available datasets showed that the developed ensemble learning methodologies and deep learning methods outperformed the corresponding benchmarks. The accurate forecasts and detection are valuable to renewable integration, power system reliability, and building integration into the smart grid.

## References

[1] Z. Huang, H. Luo, D. Skoda, T. Zhu, Y. Gu, E-sketch: gathering large-scale energy consumption data based on consumption patterns, 2014 IEEE International Conference on Big Data (Big Data), IEEE, 2014, . 656–665.
[2] C. Ramos, C.-C. Liu, AI in power systems and energy markets, IEEE Intell. Syst. 26 (2) (2011) 5–8.
[3] D. Ernst, M. Glavic, L. Wehenkel, Power systems stability control: reinforcement learning framework, IEEE Trans. Power Syst. 19 (1) (2004) 427–435.
[4] T.I. Ahamed, P.N. Rao, P. Sastry, A reinforcement learning approach to automatic generation control, Electr. Power Syst. Res. 63 (1) (2002) 9–26.
[5] T. Yu, J. Liu, K. Chan, J. Wang, Distributed multi-step Q (λ) learning for optimal power flow of large-scale power grids, Int. J. Electr. Power 42 (1) (2012) 614–620.
[6] J.M. Malof, K. Bradbury, L.M. Collins, R.G. Newell, Automatic detection of solar photovoltaic arrays in high resolution aerial imagery, Appl. Energy 183 (2016) 229–240.
[7] Y. He, G.J. Mendis, J. Wei, Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism, IEEE Trans. Smart Grid 8 (5) (2017) 2505–2516.
[8] Y. Wang, M. Liu, Z. Bao, Deep learning neural network for power system fault diagnosis, 2016 35th Chinese Control Conference (CCC), IEEE, 2016, . 6678-6683.
[9] L. Cheng, T. Yu, A new generation of AI: A review and perspective on machine learning technologies applied to smart energy and electric power systems, Int. J. Energy Res. 43 (6) (2019) 1928–1973.
[10] C. Feng, J. Zhang, Wind power and ramp forecasting for grid integration, Advanced Wind Turbine Technology, Springer, 2018, pp. 299–315.
[11] C. Feng, M. Sun, M. Cui, E.K. Chartan, B.-M. Hodge, J. Zhang, Characterizing forecastability of wind sites in the United States, Renew. Energy 133 (2019) 1352–1365.
[12] L. Cibulka, M. Brown, L. Miller, A. Meier, User requirements and research needs for renewable generation forecasting tools that will meet the need of the CAISO and utilities for 2020, in: CIEEA White Paper Report, 2012.
[13] M. Henderson *et al.*, ISO New England wind integration study, 2011 IEEE Power and Energy Society General Meeting, IEEE, 2011, . 1–6.
[14] J. Wilczak, et al., The Wind Forecast Improvement Project (WFIP): a public–private partnership addressing wind energy forecast needs, Bull. Am. Meteorological Soc. 96 (10) (2015) 1699–1718.
[15] US Department of Energy, WindView: An Open Platform for Wind Energy Forecast Visualization. Available from: <https://www.energy.gov/eere/wind/renewable-systems-integration>, 2015.

[16] US Department of Energy, Watt-Sun: a multi-scale, multi-model, machine-learning solar forecasting technology. Available from: <https://www.energy.gov/eere/solar/watt-sun-multi-scale-multi-model-machine-learning-solar-forecasting-technology>, 2012.
[17] US Department of Energy, Solar uncertainty management and mitigation for exceptional reliability in grid operations. Available from: <https://www.energy.gov/eere/solar/solar-forecasting-2>, 2017.
[18] L. Zhang, G. Wang, G.B. Giannakis, "Real-time power system state estimation and forecasting via deep neural networks," arXiv preprint arXiv:1811.06146, 2018.
[19] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, N.D. Sidiropoulos, Learning to optimize: training deep neural networks for interference management, IEEE Trans. Signal Process. 66 (20) (2018) 5438–5453.
[20] A.S. Zamzam, X. Fu, N.D. Sidiropoulos, Data-driven learning-based optimization for distribution system state estimation, IEEE Trans. Power Syst. (2019).
[21] P.N.P. Barbeiro, J. Krstulovic, H. Teixeira, J. Pereira, F.J. Soares, J.P. Iria, State estimation in distribution smart grids using autoencoders, IEEE Eighth International Power Engineering and Optimization Conference Langkawi, Malaysia, IEEE, 2014, . 358–363.
[22] R.C.B. Hink, J.M. Beaver, M.A. Buckner, T. Morris, U. Adhikari, S. Pan, Machine learning for power system disturbance and cyber-attack discrimination, Seventh International Symposium on Resilient Control Systems (ISRCS), Denver, CO, IEEE, 2014, . 1–8.
[23] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, Z. Han, Detecting stealthy false data injection using machine learning in smart grid, IEEE Syst. J. 11 (3) (2014) 1644–1652.
[24] J. Brooks, S. Kumar, S. Goyal, R. Subramany, P. Barooah, Energy-efficient control of under-actuated HVAC zones in commercial buildings, Energy Build. 93 (2015) 160–168.
[25] C.D. Korkas, S. Baldi, I. Michailidis, E.B. Kosmatopoulos, Occupancy-based demand response and thermal comfort optimization in microgrids with renewable energy sources and energy storage, Appl. Energy 163 (2016) 93–104.
[26] A. Albert, R. Rajagopal, Smart meter driven segmentation: What your consumption says about you, IEEE Trans. Power Syst. 28 (4) (2013) 4019–4030.
[27] J. Torriti, Demand side management for the European supergrid: occupancy variances of European single-person households, Energy Policy 44 (2012) 199–206.
[28] J. Zeng, W. Qiao, Support vector machine-based short-term wind power forecasting, 2011 IEEE/PES Power Systems Conference and Exposition, IEEE, 2011, . 1–8.
[29] C. Wan, Z. Xu, P. Pinson, Z.Y. Dong, K.P. Wong, Probabilistic forecasting of wind power generation using extreme learning machine, IEEE Trans. Power Syst. 29 (3) (2013) 1033–1044.
[30] J.W. Taylor, P.E. McSharry, R. Buizza, Wind power density forecasting using ensemble predictions and time series models, IEEE Trans. Energy Convers. 24 (3) (2009) 775–782.
[31] Y. Wang, N. Zhang, Y. Tan, T. Hong, D.S. Kirschen, C. Kang, Combining probabilistic load forecasts, IEEE Trans. Smart Grid (2018).
[32] C. Feng, et al., Short-term global horizontal irradiance forecasting based on sky imaging and pattern recognition, IEEE Power & Energy Society General Meeting, Chicago, IL, IEEE, 2017, . 1–5.
[33] M. Espinoza, J.A. Suykens, B. De Moor, Fixed-size least squares support vector machines: A large scale application in electrical load forecasting, Comput. Manage. Sci. 3 (2) (2006) 113–129.

[34] C. Feng, E.K. Chartan, B.-M. Hodge, J. Zhang, Characterizing time series data diversity for wind forecasting, The Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, ACM, 2017, . 113–119.
[35] J. Xia, P. Zhao, Y. Dai, Neuro-fuzzy networks for short-term wind power forecasting, 2010 International Conference on Power System Technology, IEEE, 2010, . 1–5.
[36] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, Z. Yan, A review on the forecasting of wind speed and generated power, Renew. Sustain. Energy Rev. 13 (4) (2009) 915–920.
[37] C. Voyant, et al., Machine learning methods for solar radiation forecasting: a review, Renew. Energy 105 (2017) 569–582.
[38] A.R. Khan, A. Mahmood, A. Safdar, Z.A. Khan, N.A. Khan, Load forecasting, dynamic pricing and DSM in smart grid: a review, Renew. Sustain. Energy Rev. 54 (2016) 1311–1322.
[39] M. Landry, T.P. Erlinger, D. Patschke, C. Varrichio, Probabilistic gradient boosting machines for GEFCom2014 wind forecasting, Int. J. Forecast. 32 (3) (2016) 1061–1066.
[40] M. Sun, C. Feng, J. Zhang, Aggregated probabilistic wind power forecasting based on spatio-temporal correlation, IEEE Power & Energy Society General Meeting, Atlanta, GA, IEEE, 2019, . 1–5.
[41] Y. Zhang, J. Wang, GEFCom2014 probabilistic solar power forecasting based on *k*-nearest neighbor and kernel density estimator, 2015 IEEE Power & Energy Society General Meeting, IEEE, 2015, . 1–5.
[42] S. Lou, D.H. Li, J.C. Lam, W.W. Chan, Prediction of diffuse solar irradiance using machine learning and multivariable regression, Appl. Energy 181 (2016) 367–374.
[43] H.-z Wang, G.-q Li, G.-b Wang, J.-c Peng, H. Jiang, Y.-t Liu, Deep learning based ensemble approach for probabilistic wind power forecasting, Appl. Energy 188 (2017) 56–70.
[44] M. Sun, C. Feng, E.K. Chartan, B.-M. Hodge, J. Zhang, A two-step short-term probabilistic wind forecasting methodology based on predictive distribution optimization, Appl. Energy 238 (2019) 1497–1505.
[45] J. Wu, B. Zhang, K. Wang, Application of AdaBoost-based BP neural network for short-term wind speed forecast,", Power Syst. Technol. 36 (9) (2012) 221–225.
[46] C. Feng, J. Zhang, Hourly-similarity based solar forecasting using multi-model machine learning blending, IEEE Power & Energy Society General Meeting (PESGM), Portland, OR, IEEE, 2018, . 1–5.
[47] C. Feng, M. Cui, B.-M. Hodge, S. Lu, H. Hamann, J. Zhang, Unsupervised clustering-based short-term solar forecasting, IEEE Trans. Sustain. Energy (2018).
[48] C. Feng, J. Zhang, Short-term load forecasting with different aggregation strategies, ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2018, pp. 1–10.
[49] H. Liu, H.-q Tian, Y.-f Li, Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction, Appl. Energy 98 (2012) 415–424.
[50] A. Lojowska, D. Kurowicka, G. Papaefthymiou, L. van der Sluis, Advantages of ARMA-GARCH wind speed time series modeling, 2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems, IEEE, 2010, . 83–88.
[51] R. Ye, Ensemble Time Series Forecasting With Applications in Renewable Energy (Doctor of Philosophy), Nanyang Technological University, 2016.
[52] S. Wang, N. Zhang, L. Wu, Y. Wang, Wind speed forecasting based on the hybrid ensemble empirical mode decomposition and GA-BP neural network method, Renew. Energy 94 (2016) 629–636.

[53] Y. Lin, M. Yang, C. Wan, J. Wang, Y. Song, A multi-model combination approach for probabilistic wind power forecasting, IEEE Trans. Sustain. Energy 10 (1) (2018) 226–237.
[54] D. Kim, J. Hur, Short-term probabilistic forecasting of wind energy resources using the enhanced ensemble method, Energy 157 (2018) 211–226.
[55] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[56] C. Feng, M. Cui, B.-M. Hodge, J. Zhang, A data-driven multi-model methodology with deep feature selection for short-term wind forecasting, Appl. Energy 190 (2017) 1245–1257.
[57] G. Ibarra-Berastegi, J. Saénz, G. Esnaola, A. Ezcurra, A. Ulazia, Short-term forecasting of the wave energy flux: Analogues, random forests, and physics-based models, Ocean. Eng. 104 (2015) 530–539.
[58] C. Feng, J. Zhang, Reinforcement learning based dynamic model selection for short-term load forecasting, The 10th Conference on Innovative Smart Grid Technologies, IEEE, 2019, . 1–5.
[59] D. Zhang, X. Han, C. Deng, Review on the research and practice of deep learning and reinforcement learning in smart grids, CSEE J. Power Energy Syst. 4 (3) (2018) 362–370.
[60] M. Glavic, R. Fonteneau, D. Ernst, Reinforcement learning for electric power system decision and control: past considerations and perspectives, IFAC-PapersOnLine 50 (1) (2017) 6918–6927.
[61] L. Cheng, T. Yu, A new generation of AI: a review and perspective on machine learning technologies applied to smart energy and electric power systems, Int. J. Energy Res. (2019).
[62] R. Segal, A. Sharma, M. Kothari, A self-tuning power system stabilizer based on artificial neural network, Int. J. Electr. Power Energy Syst. 26 (6) (2004) 423–430.
[63] T. Senjyu, H. Fujita, Recurrent neural network supplementary stabilization controller for automatic voltage regulator and governor, Electr. Power Compon. Syst. 31 (7) (2003) 693–707.
[64] D. Chaturvedi, O. Malik, P. Kalra, Generalized neuron based power system stabilizer,", Electr. Power Compon. Syst. 32 (5) (2004) 467–490.
[65] R. Hadidi, B. Jeyasurya, Reinforcement learning based real-time wide-area stabilizing control agents to enhance power system stability, IEEE Trans. Smart Grid 4 (1) (2013) 489–497.
[66] H. Zeynelgil, A. Demiroren, N. Sengor, The application of ANN technique to automatic generation control for multi-area power system, Int. J. Electr. Power Energy Syst. 24 (5) (2002) 345–354.
[67] D. Chaturvedi, P. Satsangi, P. Kalra, Load frequency control: a generalised neural network approach, Int. J. Electr. Power Energy Syst. 21 (6) (1999) 405–415.
[68] K. Sabahi, M. Nekoui, M. Teshnehlab, M. Aliyari, M. Mansouri, Load frequency control in interconnected power system using modified dynamic neural networks, 2007 Mediterranean Conference on Control & Automation, IEEE, 2007, . 1–5.
[69] H. Shayeghi, H. Shayanfar, Power system load frequency control using RBF neural networks based on/spl mu/-synthesis theory, IEEE Conference on Cybernetics and Intelligent Systems, 2004, IEEE, vol. 1, 2004, . 93–98.
[70] H. Li, Z. Yan, Application of Q-learning approach with prior knowledge to nonlinear AGC system,", Autom. Electr. Power Syst. 23 (2008).
[71] M. Tan, C. Han, X. Zhang, L. Guo, T. Yu, Hierarchically correlated equilibrium Q-learning for multi-area decentralized collaborative reactive power optimization, CSEE J. Power Energy Syst. 2 (3) (2016) 65–72.

[72] L. Xi, J. Chen, Y. Huang, T. Xue, T. Zhang, Y. Zhang, Smart generation control based on deep reinforcement learning with the ability of action self-optimization, Sci. Sin. Inf. 48 (10) (2018) 1430–1449.
[73] A. Lucifredi, C. Mazzieri, M. Rossi, Application of multiregressive linear models, dynamic kriging models and neural network models to predictive maintenance of hydroelectric power systems, Mech. Syst. Signal Process. 14 (3) (2000) 471–494.
[74] I. Pan, S. Das, Kriging based surrogate modeling for fractional order control of microgrids, IEEE Trans. Smart Grid 6 (1) (2014) 36–44.
[75] P.B. Luh, et al., Payment cost minimization auction for deregulated electricity markets using surrogate optimization, IEEE Trans. Power Syst. 21 (2) (2006) 568–578.
[76] M. Parvania, M. Fotuhi-Firuzabad, Demand response scheduling by stochastic SCUC, IEEE Trans. Smart Grid 1 (1) (2010) 89–98.
[77] M. Dabbaghjamanesh, A. Kavousi-Fard, S. Mehraeen, Effective scheduling of reconfigurable microgrids with dynamic thermal line rating, IEEE Trans. Ind. Electron. 66 (2) (2018) 1552–1564.
[78] M. Dabbaghjamanesh, A. Kavousifard, S. Mehraeen, J. Zhang, Z.Y. Dong, Sensitivity analysis of renewable energy integration on stochastic energy management of automated reconfigurable hybrid AC-DC microgrid considering DLR security constraint, IEEE Trans. Ind. Inf. (2019).
[79] D. Shirmohammadi, H.W. Hong, Reconfiguration of electric distribution networks for resistive line losses reduction, IEEE Trans. Power Deliv. 4 (2) (1989) 1492–1498.
[80] M.A. Kashem, V. Ganapathy, G.B. Jasmon, Network reconfiguration for load balancing in distribution networks, IEE Proc.-Gen., Transm. Distrib. 146 (6) (1999) 563–567.
[81] S. Golshannavaz, S. Afsharnia, F. Aminifar, Smart distribution grid: optimal day-ahead scheduling with reconfigurable topology, IEEE Trans. Smart Grid 5 (5) (2014) 2402–2411.
[82] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv preprint arXiv:1409.1259, 2014.
[83] J. Chaney, E.H. Owens, A.D. Peacock, An evidence based approach to determining residential occupancy and its role in demand response management, Energy Build. 125 (2016) 254–266.
[84] L.M. Candanedo, V. Feldheim, Accurate occupancy detection of an office room from light, temperature, humidity and $CO_2$ measurements using statistical learning models, Energy Build. 112 (2016) 28–39.
[85] J. Zou, Q. Zhao, W. Yang, F. Wang, Occupancy detection in the office by analyzing surveillance videos and its application to building energy conservation, Energy Build. 152 (2017) 385–398.
[86] H. Zou, Y. Zhou, J. Yang, C. Spanos, Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT, Energy Build. 174 (2018) 309–322.
[87] N. Li, G. Calis, B. Becerik-Gerber, Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations, Autom. Constr. 24 (2012) 89–99.
[88] Z. Chen, Q. Zhu, Y.C. Soh, Smartphone inertial sensor-based indoor localization and tracking with iBeacon corrections, IEEE Trans. Ind. Inf. 12 (4) (2016) 1540–1549.
[89] W. Wang, T. Hong, N. Li, R.Q. Wang, J. Chen, Linking energy-cyber-physical systems with occupancy prediction and interpretation through WiFi probe-based ensemble classification, Appl. Energy 236 (2019) 55–69.
[90] J. Zhao, B. Lasternas, K.P. Lam, R. Yun, V. Loftness, Occupant behavior and schedule modeling for building energy simulation through office appliance power consumption data mining, Energy Build. 82 (2014) 341–355.

[91] J.G. Ortega, L. Han, N. Whittacker, N. Bowring, A machine-learning based approach to model user occupancy and activity patterns for energy saving in buildings, 2015 Science and Information Conference (SAI), IEEE, 2015, . 474–482.
[92] M. Jin, R. Jia, C.J. Spanos, Virtual occupancy sensing: using smart meters to indicate your presence, IEEE Trans. Mob. Comput. 16 (11) (2017) 3264–3277.
[93] Z. Chen, R. Zhao, Q. Zhu, M.K. Masood, Y.C. Soh, K. Mao, Building occupancy estimation with environmental sensors via CDBLSTM, IEEE Trans. Ind. Electron. 64 (12) (2017) 9549–9559.
[94] H. Zou, Y. Zhou, J. Yang, C. Spanos, Towards occupant activity driven smart buildings via WiFi-enabled IoT devices and deep learning, Energy Build. 177 (2018) 12–22.
[95] Y. Wang, Q. Chen, D. Gan, J. Yang, D.S. Kirschen, C. Kang, Deep learning-based socio-demographic information identification from smart meter data,", IEEE Trans. Smart Grid (2018).
[96] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUs), arXiv preprint arXiv:.07289, 2015.
[97] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, . 6645–6649.
[98] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, S. Santini, The ECO data set and the performance of non-intrusive load monitoring algorithms, Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, ACM, 2014, . 80–89.
[99] W. Kleiminger, C. Beckel, S. Santini, Household occupancy monitoring using electricity meters, Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2015, . 975–986.
[100] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[101] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.