Deep Learning-based Real-time Building Occupancy Detection Using AMI Data

Cong Feng, *Student Member, IEEE*, Ali Mehmani, *Member, IEEE*, and Jie Zhang, *Senior Member, IEEE*

***Abstract*—Building occupancy patterns facilitate successful development of the smart grid by enhancing building-to-grid integration efficiencies. Current occupancy detection is limited by the lack of widely deployed non-intrusive sensors and the insufficient learning power of shallow machine learning algorithms. This paper seeks to detect real-time building occupancy from Advanced Metering Infrastructure (AMI) data based on a deep learning architecture. The developed deep learning model consists of a convolutional neural network (CNN) and a long short-term memory (LSTM) network. Specifically, a CNN with convolutional and max-pooling layers extracts spatial features in the AMI data. Then, the forward and backward dependencies within the CNN feature maps are learned by a bidirectional LSTM (BiLSTM) structure with three hidden layers. Case studies based on a publicly available dataset show that the developed CNN-BiLSTM model consistently and robustly outperforms the state-of-the-art machine learning classifiers and other advanced deep learning architectures with around 90% occupancy detection accuracy and high detection confidence.**

***Index Terms*—Deep learning, convolutional neural network, long short-term memory, smart meter, building occupancy detection.**

I. Introduction

THE BUILDING sector accounts for over 70% of the total electricity consumption in the U.S., making the building integration a critical part of the smart grid. Widespread research has been done to enhance communications between buildings and the grid, such as demand-side load monitoring [1] and forecasting [2], [3], customer socio-economic characterization [4], and demand response program design [5]. Smart buildings with occupancy-driven demand response are found to achieve significant energy savings, compared to conventional demand response programs that have no information about occupancy [6]–[8]. The occupancy information is beneficial to the demand response in several aspects, such as: (i) helping determine peak demand periods at the household level, (ii) identifying the potential of each house to participate and their possible participating time, and (iii) remotely and automatically actuating loads through advanced control strategies in smart buildings. For example, occupancy information was used to understand the consumers' attributes for tailored marketing actions in demand response management [9]. The importance of occupancy information to demand response was also emphasized in [10] by constructing residential occupancy curves of 15 European countries. Moreover, Brooks *et al.* [6] developed a real-time occupancy-based feedback control algorithm for variable air volume heating, ventilation, and air-conditioning (HVAC) systems, which saved 29%–80% energy savings for commercial buildings. Real-time occupancy was obtained through video-processing techniques for HVAC unit predictive control in [7]. In addition, a novel control mechanism was developed for joint demand response management and thermal comfort optimizaion, where the demand response control strategy was dependent on the occupancy information [8]. A more detailed of occupancy detection benefits to the demand response management can be found in [11].

Occupancy detection can be conducted through the use of various sensors, which can be roughly categorized into *intrusive* or *non-intrusive* groups based on the monitored objects. Intrusive sensors measure indoor environments, such as motional, acoustic, or climatic parameters [12]. For example, occupancy detection models with different indoor climate feature combinations were compared in [13] and generated the best results with over 99% accuracies. Moreover, Zou *et al.* [14] developed an occupancy detection method based on surveillance videos, which reached up to 95.3% accuracy. However, the extensive deployment of intrusive occupancy detection is challenging due to the high installation cost, additional operation requirements (e.g., the illumination condition for cameras), and privacy concerns. On the other hand, non-intrusive occupancy detection relies on infrastructure sensors that monitor parameters like WiFi [15], Bluetooth [16], and radio-frequency identification (RFID) [17]. For example, a WiFi-based occupancy classification system provided detection with a 72.7% accuracy, which helped save 26.4% of energy consumption in cooling and ventilation demands [18]. In addition, an RFID-based occupancy detection system tracked stationary and mobile occupants with corresponding accuracies of 88% and 62%, respectively [17]. Compared to intrusive approaches, the non-intrusive methods raise fewer privacy issues. Nevertheless, they still suffer from possibly unsatisfactory accuracies, low infrastructure/device coverage, and extra occupant participation. The limitations of both intrusive and non-intrusive

Manuscript received July 10, 2019; revised October 14, 2019 and February 4, 2020; accepted March 16, 2020. Date of publication March 20, 2020; date of current version August 21, 2020. Paper no. TSG-00979-2019. *(Corresponding author: Jie Zhang.)*

Cong Feng and Jie Zhang are with the Department of Mechanical Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: cong.feng1@utdallas.edu; jiezhang@utdallas.edu).

Ali Mehmani is with the Department of Core Research, Prescriptive Data Company, New York, NY 10019 USA (e-mail: amehmani@prescriptivedata.io).

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

Digital Object Identifier 10.1109/TSG.2020.2982351

sensors provide an opportunity to infer building occupancy from Advanced Metering Infrastructure (AMI) data. For example, a genetic programming-based feature engineering technique was used to create machine learning inputs from AMI data to detect occupancy of a large-scale dataset with around 90% accuracy [19]. However, AMI-based occupancy detection is still limited compared to other sensor-based detection.

The growing penetration of sensors has enabled the development of data-driven machine learning models for occupancy detection. For example, a support vector machine (SVM) was developed to detect occupancy patterns and achieved over 80% accuracy, outperforming the hidden Markov model (HMM) and k-Nearest Neighbours (kNN) model [20]. Zhao *et al.* [21] compared a collection of machine learning models in occupant behavior detection, including a decision tree (DT), an SVM, and a Bayes network, and found DT was superior to other models. Moreover, the random forest (RF) model was recognized as a better model than HMM and SVM in [22] with around 90% accuracy. While shallow machine learning models have been extensively adopted, deep learning is far from fully explored in occupancy detection problems. Compared to shallow learning methods, deep learning captures patterns in the raw data through representation learning without heavey feature engineering [23]. For example, an autorencoder long-term recurrent convolutional network was developed to identify the occupant activity with WiFi-enabled Internet of Things devices [24].

To bridge the gap in occupancy detection using widely available AMI data, in this paper we propose a deep learning architecture by sequentially stacking a CNN and a bidirectional long short-term memory network (BiLSTM). CNN is capable of learning local spatial features from the input but lacks the ability to learn sequential correlations while recurrent neural networks (RNNs) are specialized for temporal modeling but unable to extract features in a parallel manner. The developed CNN-BiLSTM architecture is expected to capture both spatial and contextual representations of the AMI data. The main contribution this paper is developing a deep learning model that combines CNN and LSTM architectures, which performs occupancy detection from AMI data without heavy feature engineering.

The remainder of the paper is organized as follows. Section II formulates the occupancy detection problem and proposes the CNN-BiLSTM architecture for AMI-based detection. Experimental setups, including the data, benchmark models, and the training strategies, are described in Section III. Section IV analyzes the results. Section V discusses the sentivity and applicability of the model. Section VI concludes the paper.

II. Methodology

The real-time occupancy information is unintrusively inferred from AMI data based on the developed CNN-BiLSTM model. This section first formulates the occupancy detection as a binary classification problem. The details of the two cornerstone components, i.e., CNN and LSTM, are concretely introduced. At last, the overall occupancy detection framework is presented.

A. Problem Formulation

In this paper, we seek to identify the real-time occupancy condition, $\mathbf{y} \in \mathbb{R}^{N\times 1}$, of a house from its AMI data, $\mathbf{X} \in \mathbb{R}^{N\times F}$ by sequentially using a CNN model and an LSTM model [25]:

$$\hat{\mathbf{y}} = F(\mathbf{X}, \mathbf{W}) = F_R\{F_C(\mathbf{X}, \mathbf{W}_C), \mathbf{W}_R\} \tag{1}$$

where N and F respectively denote the sample size and feature dimension; $\mathbf{y}$ and $\hat{\mathbf{y}}$ are actual label and detected output, respectively; F, F_C, and F_R stand for the occupancy detection model and its CNN and LSTM components, respectively; $\mathbf{W}$, $\mathbf{W}_C$, and $\mathbf{W}_R$ are the trainable parameters in the developed CNN-BiLSTM, CNN, and LSTM models, respectively. Note that the CNN-BiLSTM model is trained in a supervised learning manner. Therefore, both the AMI data and occupancy labels are required in the training stage, but only AMI data is needed in the detection stage. However, the occupancy labels can be obtained through transfer learning and unsupervised learning in real-world applications. The applicability of this work will be discussed in Section V.

Occupancy detection is a binary classifiction problem, in which the house could be either occupied or vacant, $y \in \{0, 1\}$, at every timestamp (each sample). Therefore, the weighted binary cross-entropy is designed as the loss function, which considers the imbalanced labels:

$$J(\mathbf{W}) = -\frac{1}{N}\sum_{n=1}^{N}\left[\omega y_n \log\left(\hat{y}_n\right) + (1-\omega)(1-y_n)\log\left(1-\hat{y}_n\right)\right] \tag{2}$$

where ω is the binary cross-entropy weight, defined as: $\omega = P_Y[y = 0|y \in \mathbf{y}]$. The objective is to optimize parameters of the developed network by minimizing the loss function in an end-to-end manner (the optimizatin method will be introduced in Section III-C).

B. Convolutional Neural Network (CNN)

CNN architectures have one or multiple feature learning blocks (FLB) that consist of a convolutional layer and a max-pooling layer. A convolutional layer (indexed by l and represented by green blocks in Fig. 1) contains a filter bank with $D^{(l+1)}$ filters. Each filter function is convoluted with the input to construct a set of feature maps: $\mathbf{Z}^l = \mathbf{W}^l * \mathbf{X}^l + \mathbf{b}^l$.

After convolutions, an element-wise rectified linear unit (ReLU) activation function is used in each FLB to add nonlinearity to the network. The ReLU function is selected because of its computational efficiency, better convergence, superior performance, and the amelioration of vanishing gradients compared to others, such as the sigmoid function [26]. The ReLU function is applied to every element of its input (which is the feature maps of the last convolutional layer).

The last layer of an FLB is a max-pooling layer (represented by red blocks in Fig. 1), which is used to achieve more translation invariance during spatial representation learning. A unified max-pooling layer with a non-overlapping moving

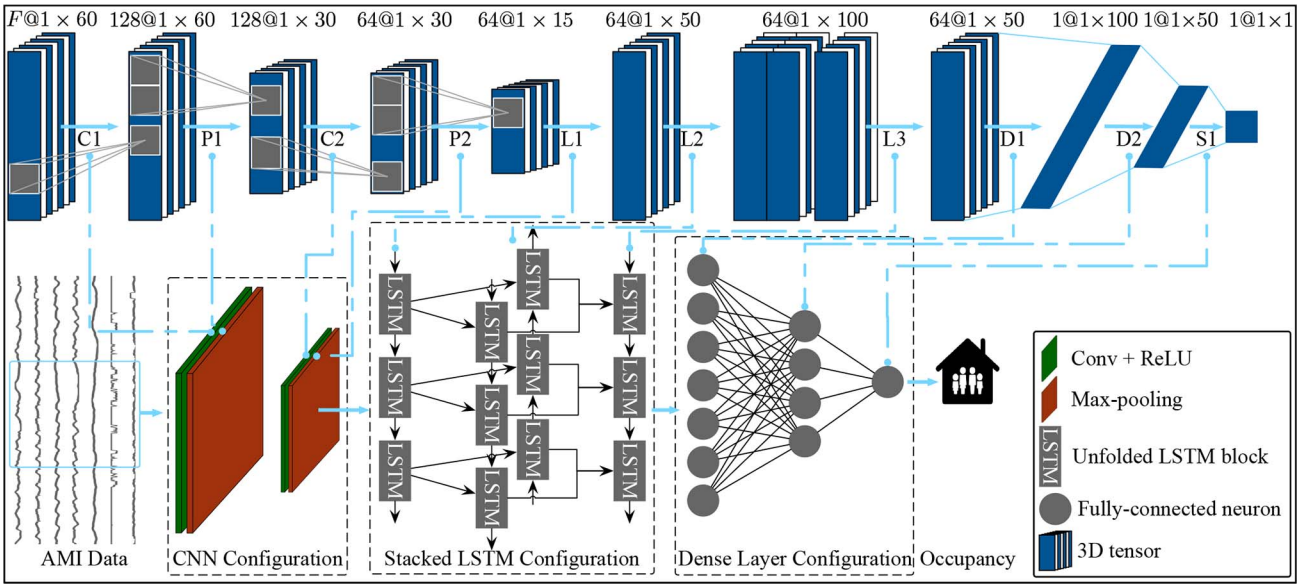


Fig. 1. The developed end-to-end occupancy detection framework based on CNN-BiLSTM and its tensor manipulation.

window (size 2×1 and stride 2) is applied to sub-sample the convoluted feature maps by a factor of 2.

C. Long Short-Term Memory Network (LSTM)

Recurrent neural networks (RNNs) have been proved to be efficient in processing temporal data. Nevertheless, conventional RNNs suffer from two limitations, known as the vanishing gradient problem during back-propagation and the inadequate modeling of backward dependencies [27], [28].

To tackle the first issue, the LSTM network, an RNN architecture with gated regulators, is deployed. In this paper, an LSTM network is stacked upon the CNN component to learn the contextual dependencies, shown in the second dashed box at the bottom of Fig. 1. Each LSTM block is composed of a memory cell, a forget gate, an input gate, and an output gate. Denoting the activation vectors of the forget, input, output gates and the memory cell as $\boldsymbol{f}$, $\boldsymbol{i}$, $\boldsymbol{o}$, and $\boldsymbol{c}$, the nodal connections and tensor operations in a forward LSTM hidden layer are expressed as:

$$\boldsymbol{f}_t = \sigma\left(\mathbf{W}_f\left[\boldsymbol{h}_{t-1}, \mathbf{X}_t\right] + \mathbf{b}_f\right) \tag{3a}$$
$$\boldsymbol{i}_t = \sigma\left(\mathbf{W}_i\left[\boldsymbol{h}_{t-1}, \mathbf{X}_t\right] + \mathbf{b}_i\right) \tag{3b}$$
$$\tilde{\boldsymbol{c}}_t = tanh\left(\mathbf{W}_c\left[\boldsymbol{h}_{t-1}, \mathbf{X}_t\right] + \mathbf{b}_c\right) \tag{3c}$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t \tag{3d}$$
$$\boldsymbol{o}_t = \sigma\left(\mathbf{W}_o\left[\boldsymbol{h}_{t-1}, \mathbf{X}_t\right] + \mathbf{b}_o\right) \tag{3e}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t * tanh(\boldsymbol{c}_t) \tag{3f}$$

where $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_c$, and $\mathbf{W}_o$ are the input weight matrices, while $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_c$, and $\mathbf{b}_o$ are the cooresponding bias vectors; $\sigma(\cdot)$ and $tanh(\cdot)$ are the logistic sigmoid and hyperbolic tangent activation functions; $\boldsymbol{h}$ is the hidden state, which is also output of the LSTM hidden layer; $\tilde{\boldsymbol{c}}$ is the new state candidate vector; and the bracket is the concatenation operator.

Both forward and backward dependencies should be included in the time series forecasting, especially in time series with evident periodicities, such as AMI data [29]. This is because that the backward dependencies captured in reverse-chronologically ordered data provide unique and useful information that the forward dependencies can not provide. The backward features can possibly improve the model perofrmance, as is proved in many fields [29]. However, most recurrent models (time series models or RNNs) can only deal with unidirectional dependencies. To solve this issue, a bidirectional LSTM (BiLSTM) layer is included in the LSTM configuration to capture both forward and backward dependencies in the AMI data. The node connections and tensor calculations in a BiLSTM are almost identical to a unidirectional LSTM, except for the processing directions. Specifically, the operations within a BiLSTM have two directions, such as:

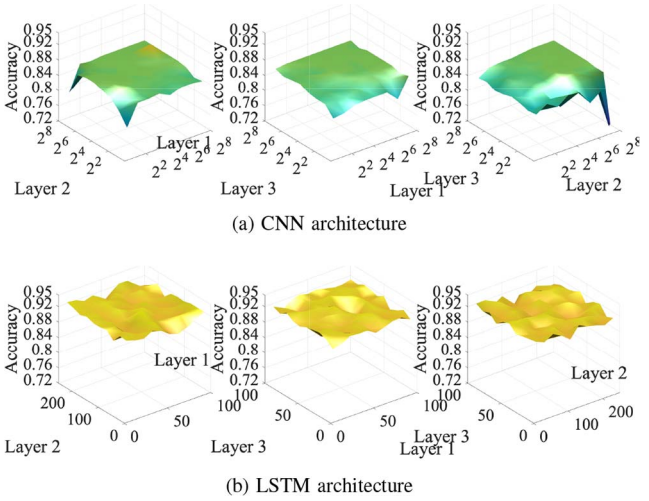
$$\overrightarrow{\boldsymbol{f}}_t = \sigma\left(\overrightarrow{\mathbf{W}}_f\left[\overrightarrow{\boldsymbol{h}}_{t-1}, \overrightarrow{\mathbf{X}}_t\right] + \overrightarrow{\mathbf{b}}_f\right) \tag{4a}$$
$$\overleftarrow{\boldsymbol{f}}_t = \sigma\left(\overleftarrow{\mathbf{W}}_f\left[\overleftarrow{\boldsymbol{h}}_{t+1}, \overleftarrow{\mathbf{X}}_t\right] + \overleftarrow{\mathbf{b}}_f\right) \tag{4b}$$

where $\rightarrow$ and $\leftarrow$ denote forward and backward operations, respectively. Two hidden state vectors, $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$, are generated independently and concatenated to the final hidden state vector in the BiLSTM layer:

$$\boldsymbol{h}_t = \left[\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t\right]. \tag{5}$$

D. The Overall CNN-BiLSTM Occupancy Detection Framework

The overall framework of the developed CNN-BiLSTM architecture and its tensor manipulations are illustrated in Fig. 1. The architecture is built between AMI data and house occupancy in an end-to-end manner. A batch of AMI data is first convoluted to spatial feature maps through the CNN configuration of CNN-BiLSTM. A VGGNet-like (i.e., a popular CNN architecture) CNN structure is constructed due to the outperformance of VGGNet over other CNN configurations [30]. As shown in the bottom left dashed box of Fig. 1, the CNN configuration has two sequentially stacked FLBs and a total of



(a) CNN architecture

(b) LSTM architecture

Fig. 2. Benchmark parameter, hyperparameter, and topology grid search optimization with 10-fold cross-validation. Note that the GP kernel is optimized automatically in the training process, therefore, is not included in the this optimization.

four layers (two convolutional layers and two max-pooling layers). The filter numbers of the two FLBs are 128 and 64. The topology and hyperparameters of the designed CNN architecture are determined by grid search optimization with 10-fold cross validation. Specifically, three-layer CNNs are built with different combinations of filters. The CNN with 128, 64, and 0 filters in the first, second, and third layer outperforms other architectures. The performance of models with various topologies and hyperparameters is shown in Fig. 2a. There are a total of 24 3-dimensional figures in the CNN optimization, however, only 3 figures with the best third dimension value are shown in Fig. 2a. The designed CNN part is expected to extract high-level abstract spatial features from the AMI data.

The extracted abstract CNN features are fed into the LSTM configuration. Although LSTM with one hidden layer might be able to capture the contextual patterns in the AMI data, increasing the depth of architecture by vertically stacking multiple hidden layers can enhance the performance [31]. Therefore, three LSTM blocks are stacked to operate the hidden state at different timescales and learn hierarchical representations of convolutional time series in the AMI data. As shown in the bottom middle dashed box of Fig. 1, the first LSTM layer extracts temporal features in the form of hidden state from the previous convolutional feature maps. Then, a BiLSTM layer considers both forward and backward dependencies in the configuration, which are combined by the last LSTM layer. The numbers of neurons in the first, second, and third LSTM layers are 50, 100, and 50, respectively. Similar to CNN, the topology and hyperparameter optimization of the LSTM is performed by the grid search with 10-fold cross validation on top of the optimal architecture. Only three LSTM layers are included in the LSTM architecture, since by adding more layers, the validation accuracy results in diminishing returns. Three 3-dimensional figures with the best third dimension value are shown in Fig. 2b. It is first observed that by adding the LSTM components, the validation accuracy is significantly improved. In addition, the hyperparameters of the LSTM layers have a stronger impact on the model performance than those of CNN layers.

TABLE I
HYPERPARAMETERS OF LAYERS IN THE DEVELOPED CNN-BILSTM MODEL

Layer	Type	Hyperparameter	Trainable parameters
C1	Convolution	Input size: $60\times F$ Filter size: 3 Filter number: 128	11,648
P1	Pooling	Window size: $2\times$ 128 Stride: 2	0
C2	Convolution	Input size: 30×128 Filter size: 3 Filter number: 64	24,640
P2	Pooling	Window size: $30\times$ 64 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	5,100
D2	Dense	Neurons: 50	5,050
S2	Classification	Neurons: 1	51

The last part is a dense layer configuration, which consists of two fully-connected layers and a classification layer. The hidden states of the last LSTM layer are flattened and fed into the first dense layer, whose outputs are input to the second dense layer:

$$\mathbf{Z}^l = \mathbf{W}^l \cdot \mathbf{X}^l + \mathbf{b}^l \tag{6}$$

where all the inputs are transmitted to the output, which is passed to the next dense layer. The last layer of the framework is a sigmoid classification layer:

$$\mathbf{Z}^l = \frac{1}{1 + e^{-\mathbf{W}^l \cdot \mathbf{X}^l + \mathbf{b}^l}} \tag{7}$$

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

$$\hat{y} = H(z - th) \tag{8}$$

where $H(\cdot)$ is the Heaviside step function and $th = 0.5$ is the threshold value. Please note the network configuration and hyperparameters are empirically determined by the best validation accuracy.

The hyperparameters of each layer of the developed CNN-BiLSTM framework are listed in Table I. The framework with hyperparameters in Table I has 240,489 trainable parameters (2.5 MB), which is a relatively small network, compared to other architectures (e.g., 16-layer VGGNet has 528 MB weights). Parameters in each layer are initialized by the Xavier method, where biases are initialized as zeros and the initialized weights conform a Gaussian distribution [32]. The tensor processing is illustrated at the top of Fig. 1. To avoid overfitting, 20% of neurons are randomly dropped in the convolutional, LSTM, and dense layers.

III. EXPERIMENTS

This section introduces the numerical experiments for AMI-based occupancy detection, including the experimental dataset description, benchmark model selection, deep learning hyperparameter settings, and the model training process.

TABLE II
DATA SUMMARY OF CASE STUDIES

Case notation	Data dimension (N, W, F)	Label (occupied/vacant)
Case1	(935, 60, 30)	774/161
Case2	(1,103, 60, 30)	831/272
Case3	(1,079, 60, 24)	771/308
Case4	(1,367, 60, 29)	1,044/323

A. Data Description

The dataset we used to validate the developed CNN-BiLSTM model is the Electricity Consumption and Occupancy (ECO) dataset.[1] ECO dataset is the largest publicly available dataset with both AMI and occupancy data [33]. The data contains general parameters like most AMI datasets, such as current, voltage, phase angle, and power. The fine-grained data were collected in five houses in summer and winter periods with one-second interval (aggregated from data with 1 Hz frequency). Basic features were extracted from the real power of each phase and the sum of all phases within every minute, including the minimum, maximum, arithmetic average, standard deviation, sum of absolute difference, autocorrelation at lag 1, and range. Please refer to [34] for more EKO AMI data and its feature extraction. Then, the minute data is flattened every hour in the CNN input format with data width $W = 60$. To ensure the successful training of the classification models, the ECO data were quality-controlled with two criteria: (i) the data length should be more than 900, and (ii) both occupancy labels (i.e., occupied and vacant conditions) should be more than 10% of the total occupancy data. Four periods of data from three houses qualified both criteria, which are summarized in Table II and used for case studies. We believe the general format and similar quality issues of the ECO dataset ensure the applicability of the developed method to other datasets. The data of each case study was divided into training, validation, and testing datasets by a ratio of 3:1:1, which is used to train one CNN-BiLSTM.

B. Benchmark Models

A collection of six state-of-the-art machine learning classifiers are used to compete with the developed CNN-BiLSTM model, which are a kNN model [35], an SVM with linear kernel [35], a Gaussian process model (GP) [36], a RF [13], a multi-layer perceptron classifier (MLP) [37], and an adaptive boosting model (AdaBoost) with decision trees as base learners [38]. The benchmark model pool covers non-parametric model (kNN), kernel-based model (SVM), feedforward neural network (MLP), and ensemble learning models (RF and AdaBoost), which are widely used in occupancy detection and other classification problems. The parameters, hyperparameters, and topologies of benchmark models were optimized by the 10-fold cross-validation grid search based on the training and validation datasets. The optimization processes are visualized in Fig. 3 and the results are listed in Table III, where definitions of parameters can be found in `scikit-learn` library in Python [39].

[1] https://www.vs.inf.ethz.ch/res/show.html?what=eco-data

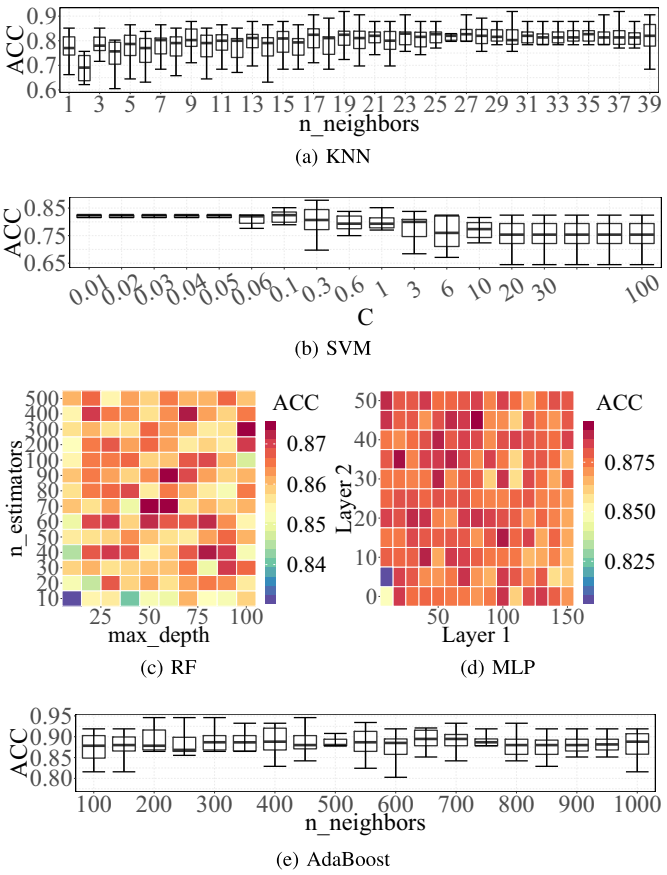


Fig. 3. Benchmark parameter, hyperparameter, and topology grid search optimization with 10-fold cross-validation. Note that the GP kernel is optimized automatically in the training process, therefore, is not included in the this optimization.

TABLE III
OPTIMAL BENCHMARK MODEL HYPERPARAMETER

Model	Hyperparameters
kNN	`n_neighbors=27`
SVM	`C=0.01`
GP	`kernel=1.0 × RBF(1.0)`
RF	`max_depth=100, n_estimators=300`
MLP	`Layer 1=80, Layer 2=45`
AdaBoost	`n_estimators=400`

In addition to the machine learning models, the developed CNN-BiLSTM are also benchmarked against several deep learning models to verify the effectiveness of combined CNN-BiLSTM and BiLSTM architectures. The deep learning benchmarks include CNN, LSTM, CNN-LSTM without BiLSTM, and convolutional LSTM (CLSTM). Their topologies (i.e., layer number) and hyperparameters (i.e., neurons in each layer) are the same as CNN-BiLSTM model.

C. Deep Learning Training

The CNN-BiLSTM is trained by mini-batch stochastic gradient descent (SDG). SDG minimizes the objective function $J(\mathbf{W})$ in Eq. (2) by updating the parameters in the opposite direction of its gradients with respect to the parameters. The complete training dataset is passed forward and backward through the network with 100 epochs. Mini-batches

($\mathbf{B} \coloneqq [\mathbf{X}, \mathbf{y}]$), with a batch size of 30, are randomly generated to shuffle the data order in every epoch. In each iteration, gradients are calculated by averaging the gradients over the mini-batch. The mini-batch SDG enhances the convergence stability and efficiency, since it reduces the variance of parameter updates and utilizes the matrix optimization techniques. 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:

$$\mathbf{V}_{i+1} = \gamma \mathbf{V}_i + \eta_{i+1} \nabla_{\mathbf{W}} J(\mathbf{W}; \mathbf{B}) \tag{9a}$$

$$\mathbf{W}_{i+1} = \mathbf{W}_i - \mathbf{V}_{i+1} \tag{9b}$$

where i is the iteration index; $\mathbf{V}$ is the weight update matrix; $\gamma = 0.9$ is the momentum and η is the learning rate.

IV. Results and Discussion

The case studies are conducted on a workstation with an Intel Xeon E5-2603 1.6 GHz CPU and an NVIDIA TITAN V GPU. The CNN-BiLSTM and benchmark models are implemented using the `Keras` library with `Tensorflow` backend and the `scikit-learn` library in Python version 3.6. Since randomness exists in the experiments (e.g., sampling, SDG, etc.), experiments are repeated 10 times with the same random number generator seed to improve the reproducibility and consistency. It took around 3.7ms to detect the occupancy at a time step, which is applicable for real-time detection.

A. Evaluation Metrics

As a binary classification problem, the occupancy detection results can be evaluated based on a 2×2 confusion matrix. The four elements in the confusion matrix are: (i) true positive (TP), denoting the count that a house is actually occupied and is detected occupied, (ii) false positive (FP), denoting the count that a house is actually vacant but is detected occupied, (iii) true negative (TN), denoting the count that a house is actually vacant and is detected vacant, and (iv) false negative (FN), denoting the count that a house is actually occupied but is detected vacant.

Five metrics are calculated based on the confusion matrix, including accuracy (*ACC*), sensitivity (*SNS*, also known as true positive rate, denoted as TPR, or recall), specificity (*SPC*, also known as true negative rate), precision (*PRC*), and F1 score ($F1$) [40], [41]. These five metrics quantify the effectiveness of occupancy detection from different perspectives. Specifically, *ACC* is the overall correctness of occupancy detections. *SNS* measures the proportion of actual occupied conditions that are correctly detected as such and *SPC* measures the proportion of actual vacant conditions that are correctly detected as such. *PRC* indicates the success probability of detecting a correct occupied condition. $F1$ takes several metrics into consideration in the imbalanced classification problems. A larger *ACC*, *SNS*, *SPC*, *PRC*, or $F1$ value indicates a better occupancy detection.

The occupancy condition is determined based on a threshold $th = 0.5$, which has a significant impact on the detection performance. Therefore, another set of metrics are used to assess the classifier performance over the entire operating range, which are the receiver operating characteristic (ROC) curve and area under the ROC curve (AUC). The former one is a curve of TPR against false positive rate ($1 - SPC$, denoted as FPR) at various threshold settings, which can also be used to determine the best th.[2] 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.

B. Basic Results

Confusion matrices of one set of experiments (randomly selected) are shown in Fig. 4. Results of four cases are located in different rows and matrices of eleven models are in different columns. It is found that testing data labels could be balanced, although all four cases have imbalanced labels in the whole dataset (as listed in Table II). For example, Case2 has 115 vacant conditions and 106 occupied conditions in the testing data. The diversity of the four cases directly impacts the model performance. For example, results of the kNN model have more FP than TN in Case1 but vice versa in Case2 and Case4. Moreover, different models show distinctive learning abilities in the same case. For example, SVM, GP, and MLP accurately detect more occupied conditions, while AdaBoost is more powerful in detecting the vacant condition in Case2. It is concluded that both the dataset and the benchmark models are diverse and general enough for the comparisons.

C. CNN-BiLSTM Outperformance

The evaluation metrics are calculated based on confusion matrices for each set of experiments, such as shown in Fig. 4, averaged over 10 repeats, and listed in Table IV. Among benchmark models, every model has the chance to generate satisfactory occupancy detection, since eight out of ten benchmark models outperform others in some cases and based on some metrics. However, none of the benchmark models can always beat the others in all the four cases. Moreover, it is observed that some models make extremely assertive detections, such as GP, which leads to 100% *SNS* but 0% *SPC* in Case1, Case3, and Case4. This is due to that some models are significantly impacted by the imbalanced data and tend to assign dominant label to the detection results, as shown in Fig. 4. SVM and RF are also occasionally influenced by the imbalanced training data. Some benchmark models are not robust, which is illustrated by the nonnegligible variance of some metrics over 10 experiment repeats, such as *SPC* of RF in Case2 and *SPC* of MLP in Case4.

The developed CNN-BiLSTM model shows encouraging accuracy in both the occupied and vacant detections based on *SNS* and *SPC*. More importantly, the CNN-BiLSTM model is more accurate than benchmark models in all the four cases,

[2]Typically, the threshould th is set as 0.5 in classification problems. However, the threshould can be optimized by selecting an operating point that is close to ideal or far from random on the ROC curve, if the data has stable characteristics [42].

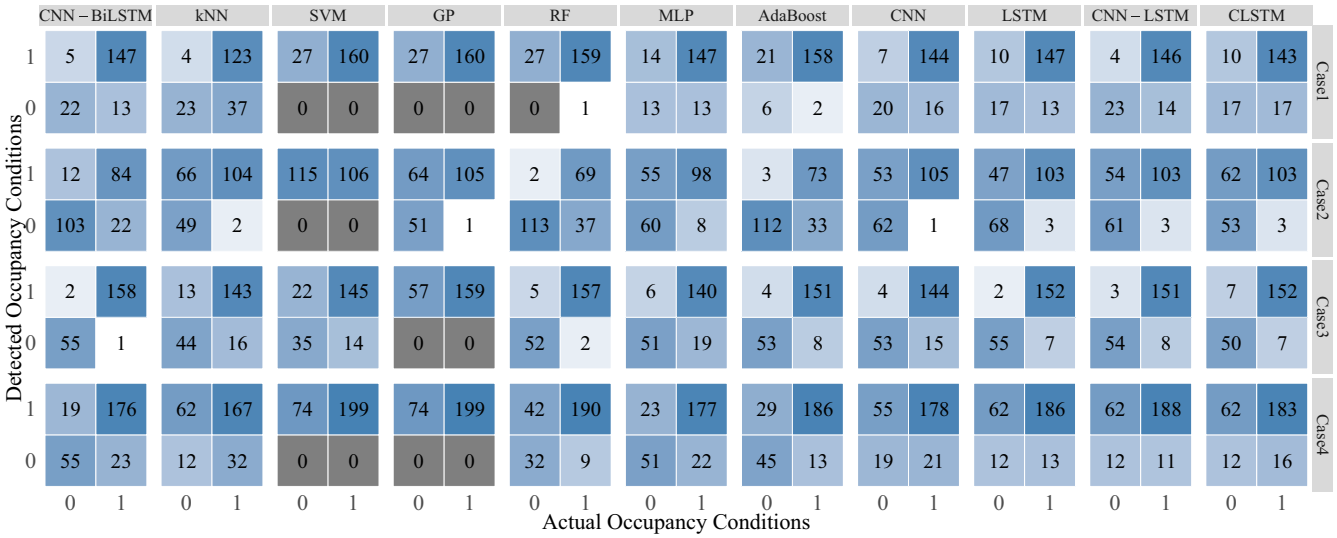


Fig. 4. Confusion matrices of the occupancy detection results. The darker color indicates a higher frequency and the grey color means a zero occurrence. Positive diagonal elements indicate right detections and negative diagonal elements show the wrong detections.

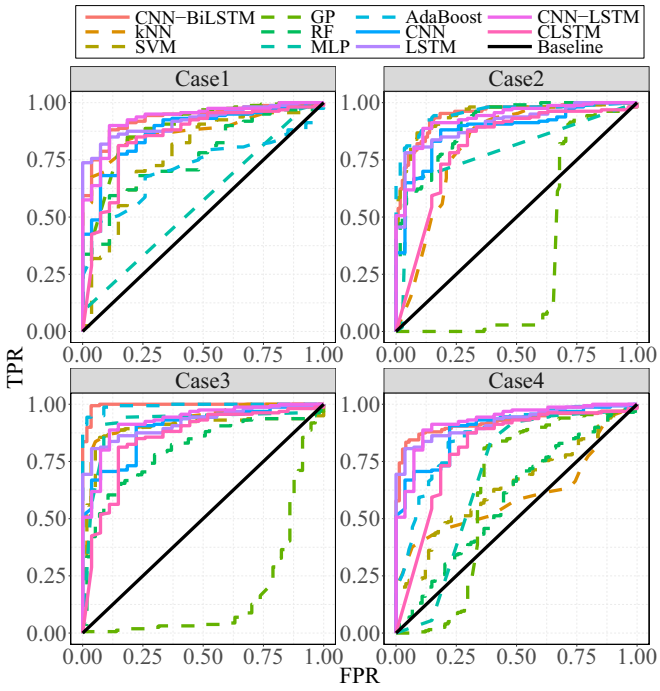
TABLE IV
EVALUATION METRICS [%] AND TRAINING TIME [MINS] OF OCCUPANCY DETECTION RESULTS

Case	Metric	Model										
		CNN-BiLSTM	kNN	SVM	GP	RF	MLP	AdaBoost	CNN	LSTM	CNN-LSTM	CLSTM
Case1	ACC	**91.34**	78.07	85.56	85.56	85.03	85.24	87.70	87.70	85.96	*90.24*	85.03
	SNS	93.13	76.88	***100***	***100***	99.38	91.50	98.75	90.78	89.84	91.09	88.75
	SPC	80.74	***85.19***	0	0	0	48.15	22.22	69.44	62.96	***85.19***	62.96
	PRC	96.63	96.85	85.56	85.56	85.48	91.27	88.27	94.63	93.5	***97.33***	93.43
	$F1$	**94.84**	85.71	92.22	92.22	91.91	91.39	93.22	92.66	91.63	*94.11*	91.02
	# Par	255,229	–	–	–	–	6,171	–	121,629	205,343	205,029	11,983,709
	Time	9.61	6.62×10^{-4}	0.05	0.13	2.89×10^{-4}	0.25	0.11	1.03	38.95	9.12	119.07
Case2	ACC	**85.25**	69.23	47.96	70.59	82.76	71.31	*83.71*	76.92	77.15	76.70	71.19
	SNS	82.36	98.11	***100***	99.06	65.09	92.92	68.87	96.70	98.11	97.41	98.74
	SPC	87.91	42.61	0	44.35	***99.04***	51.39	97.39	58.70	57.83	57.61	45.80
	PRC	86.79	61.18	47.96	62.13	***98.44***	63.86	96.05	68.37	68.40	67.98	62.67
	$F1$	**84.21**	75.36	64.83	76.36	78.36	75.67	80.22	80.08	*80.54*	80.06	76.68
	# Par	255,229	–	–	–	–	6,171	–	121,629	205,343	205,029	11,983,709
	Time	11.56	7.78×10^{-4}	0.06	0.34	3.15×10^{-4}	0.28	0.14	1.26	43.1	12.43	141.01
Case3	ACC	**98.47**	86.57	83.33	73.61	*96.16*	88.94	94.17	91.20	93.87	95.72	91.51
	SNS	**99.12**	89.94	91.19	100	*98.30*	89.25	94.59	91.35	94.65	95.75	92.24
	SPC	63.92	77.19	61.40	0	90.18	88.07	92.98	90.79	91.67	***95.61***	89.47
	PRC	87.21	91.67	86.83	73.61	96.54	95.46	97.41	96.52	96.96	***98.42***	96.07
	$F1$	88.20	90.79	88.96	84.80	97.41	92.21	95.98	93.86	95.78	***97.04***	94.10
	# Par	252,925	–	–	–	–	5,691	–	119,325	205,343	202,725	9,679,709
	Time	11.34	6.36×10^{-4}	0.05	0.14	2.83×10^{-4}	0.25	0.10	1.48	45.15	11.24	111.16
Case4	ACC	**93.13**	65.57	72.89	72.89	81.36	83.44	*84.62*	81.68	75.09	79.85	76.19
	SNS	89.50	83.92	***100***	***100***	95.58	90.50	93.47	94.47	96.98	89.95	89.45
	SPC	63.92	16.22	0	0	43.11	***64.46***	60.81	47.30	16.22	52.70	0.54
	PRC	87.21	72.93	72.89	72.89	81.88	***87.30***	86.51	82.82	75.69	83.64	80.18
	$F1$	88.20	78.04	84.32	84.32	88.20	88.85	***89.86***	88.26	85.02	86.68	84.56
	# Par	254,845	–	–	–	–	6,091	–	121,245	205,343	204,645	11,599,709
	Time	5.86	1.31×10^{-3}	0.10	0.30	3.47×10^{-4}	0.30	0.13	1.67	60.16	14.05	169.18

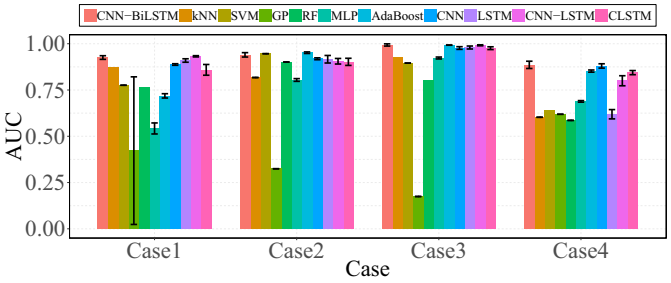
Note: **Bold values** indicate the best detection models and *italic values* indicate the best benchmark model. The darker green color indicates a better results and the darker yellow color indicates a worse results.

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. Specifically, the designed architecture in the CNN-BiLSTM effectively takes advantages of both CNN and LSTM learning abilities in capturing the spatial and temporal features in the data, revealed by its superior performance than LSTM and CNN, respectively. What's more, the additional backward temporal dependencies are shown to further enhance the occupancy detection accuracy by comparing CNN-BiLSTM and CNN-LSTM. In contrast to CLSTM, the stacked CNN and LSTM architecture of the CNN-BiLSTM is more advanced than the architecture of encoding CNN within LSTM in performing building occupancy detection. The robustness of the CNN-BiLSTM model is not only revealed in diverse cases but also shown in 10 sets of experiments.

The occupancy condition is determined by both the classification probability and the threshold, as indicated in Eq. (8).



(a) Receiver operating characteristic (ROC) curves in randomly selected experiments



(b) The Area under ROC curve (AUC) statistics of the 10 experiment repeats

Fig. 5. Speculation model performance by ROC and AUC.

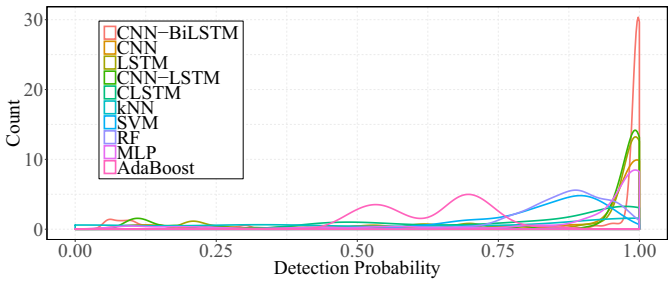


Fig. 6. Detection probability distributions of different models.

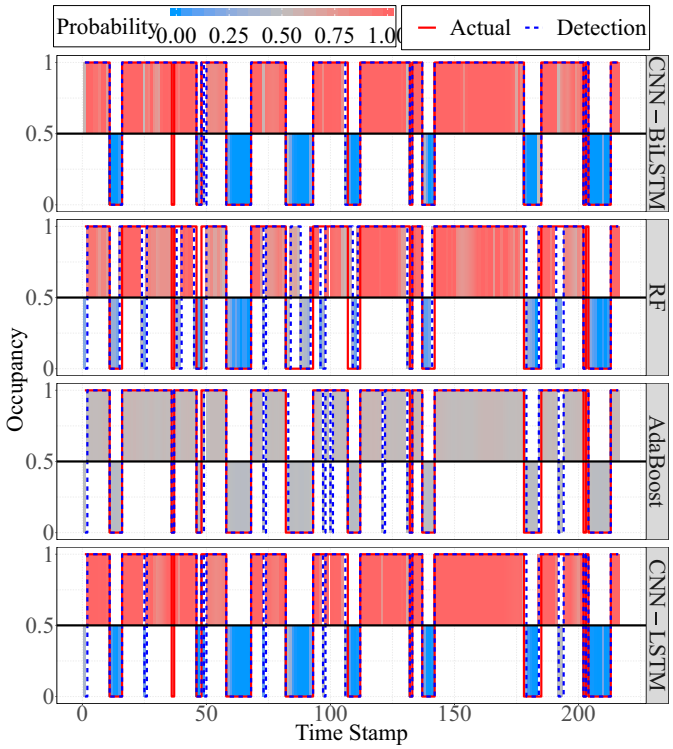
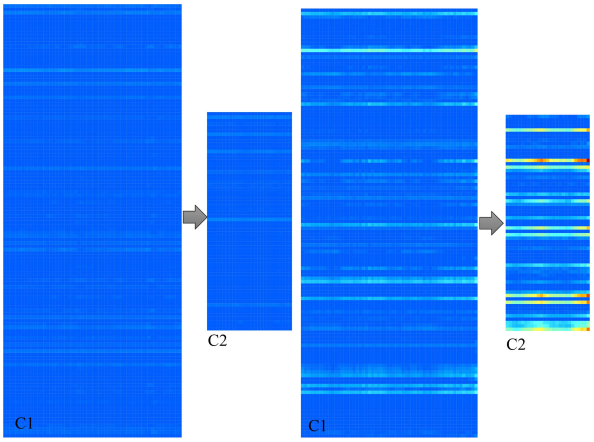


Fig. 7. The occupancy labels, detection probabilities, and results of Case3.

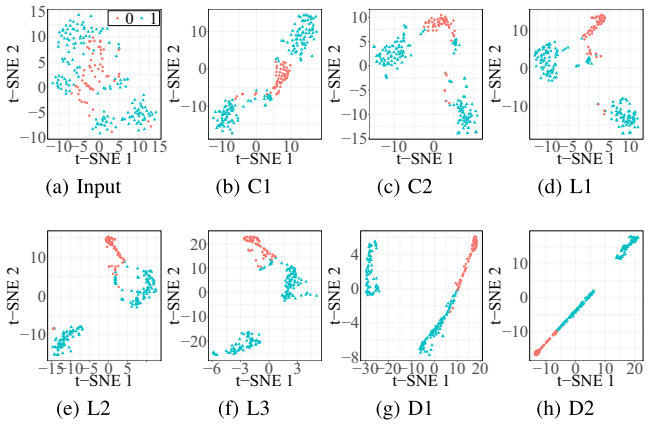
Hence, ROC and AOC are adopted to assess the model performance independent of the choice of *th*. To obtain ROC, a set of *th* values, ranging from 0 to 1, are used to determine $\hat{y}$ and calculate *TPR* and *FPR*, as shown in Fig. 5a. The perfect classifier has the ROC curve going straight up the vertical axis then along the horizontal axis. The classifier that randomly generates occupancy detection results sits on the diagonal and the classifier detects completely reverse results has a curve in the bottom left part of the ROC space. Therefore, the developed CNN-BiLSTM model shows better performance in terms of the "detectability" and robustness with different thresholds. From AUC means and variances in Fig. 5b, it is observed that the outperformance and robustness of the CNN-BiLSTM model are consistent in the 10 experimental sets. For example, compared to shallow learning models, such as kNN, GP, RF, and MLP, the CNN-BiLSTM model has larger AUC in all cases and runs. In Case 2, two models show competitive performance to the CNN-BiLSTM model, which are SVM and AdaBoost. However, considering their robustness in different cases, they are not suitable for wide applications. Moreover, CNN-BiLSTM outperforms other deep learning benchmarks, which indicates its effective architecture design.

Figure 6 illustrates the behavior difference of the models by their detection probability distributions. It is observed that deep learning models have distributions with similar shapes, whereas the CNN-BiLSTM model has more detections with probability close to 1. The detection probability distributions of shallow learning models are distinctive to CNN-BiLSTM and are various. Figure 7 digs into the reason of better occupancy detection performance of the CNN-BiLSTM model by comparing its detection probabilities and results to those of the three best benchmarks, i.e., RF, AdaBoost, and CNN-LSTM. The advantages of the CNN-BiLSTM are twofold. The first one is more accurate detection results, revealed by the smaller deviations between detections and actual conditions. Specifically, the CNN-BiLSTM 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 benchmarks generate more and longer-period



(a) A vacant case (b) An occupied case

Fig. 8. CNN layer reactions to occupancy conditions in terms of neuron activations.



(a) Input (b) C1 (c) C2 (d) L1

(e) L2 (f) L3 (g) D1 (h) D2

Fig. 9. Visualization of the feature distribution of test samples via t-SNE method.

wrong detections. The second advantage is that the CNN-BiLSTM model is more confident in the detection. Since the final occupancy detection results are determined by Eq. (8), classification probabilities that are closer to 0/1 are more confident and stable when being classified into vacant/occupied conditions. Compared to other models, such as AdaBoost, the CNN-BiLSTM model shows more confidence, which is illustrated by the darker color of the classification probability boxes. These two advantages explain the better performance of the developed CNN-BiLSTM model.

D. CNN-BiLSTM Visualization

Deep learning models are generally regarded as black boxes, since their inner operation mechanism is challenging to interpret. In this section, we explain the working logic of the CNN-BiLSTM model by digging into the neurons of important layers.

Different from typical feature engineering that is widely used in shallow learning, feature importance can not be ranked by neither filter methods nor wrapper methods in the CNN-BiLSTM. This is due to that the input of the CNN-BiLSTM model is augmented and extracted by filters in the first CNN layer rather than being fed into activation functions as in MLP. As an alternative, we explore the convolutional filters and analyze patterns they emphasize, as they look directly at the input data. It is especially interesting to show the convolutional features by investigating the reactions of the CNN feature extractors when fed with data of vacant and occupied cases. Figure 7 shows the neuron activations of the two CNN layers (i.e., C1 and C2) in two occupancy conditions. The C1 layer has output with dimension of a 128×58 (i.e., the number of filters and the number of moving windows) and the C2 layer has output with dimension of a 64×27. As seen from the figure, features extracted by several filters in the C1 layer are evidently discintive in two conditions. And these distinctions are reinforced by C2 layer filters, as shown from the brighter colors in C2 activations.

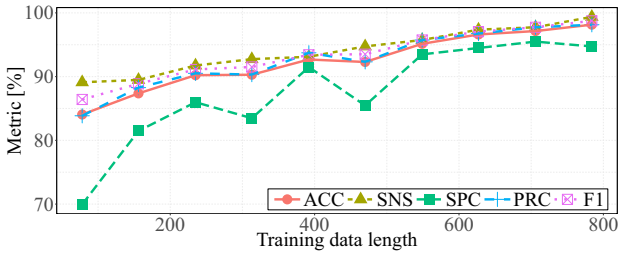
In addition, the feature learning process of the CNN-BiLSTM model is analyzed by visualizing the feature distribution in each layer (meaningless max-pooling layers are excluded) via the t-Distributed Stochastic Neighbor Embedding technique (t-SNE) [43]. The t-SNE converts high-dimensional layer outputs to two-dimensional samples for the entire Case2 testing dataset, as shown in Fig. 9. It is first observed from Fig. 9a that input samples of the two occupancy conditions are mutual included. Then, most vacant samples are able to be discriminated from occupied samples by the two convolutional layers, as shown in Figs. 9b and 9c. Moreover, the temporal learning abilities of the three LSTM layers easily isolated part of occupied samples from vacant samples. It is also worth to point out that the progressively improved classification in Figs. 9d-9f indicates effectiveness of the designed stacked BiLSTM architecture. The last two dense layers make the occupancy conditions linearly dividable. Even though the CNN-BiLSTM efficiently detects most occupancy conditions, there are still three points being mis-classified. This is possibly due to that the occupants leave the electric appliances turnning on and leave the house or the occupants go back home but without changing the electricity use patterns, which leads to similar features with counterpart conditions in the CNN-BiLSTM.

V. Sensitivity Analysis and Discussion

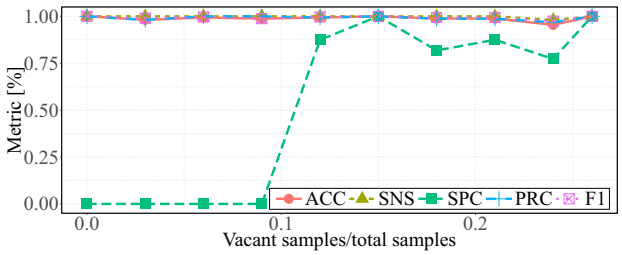
The developed CNN-BiLSTM model has been verified based on the ECO dataset, which is the largest dataset with both AMI and occupancy data. Even though the dataset is still small compared to those applied in other research, such as the global energy forecasting competition (GEFCom) dataset [44], the CNN-BiLSTM model is robust and flexible to large-scale real-world AMI data. In this section, we perform the data sensitivity analysis and discuss model applicability to AMI data without occupancy labels.

A. Sensitivity Analysis

The sensitivity analysis case studies are performed based on Case3 due to it's relevantly more balanced and larger. First, we explore the CNN-BiLSTM model's sensitivity to data length. Training data with different lengths are created, while the testing data remains the same. A total of ten cases with randomly



(a) Occupancy detection results with varying training data length



(b) Occupancy detection results with varying label imbalance

Fig. 10. Sensitivity analysis results.

selected training samples have 10%–100% data, compared to the original training dataset. Note that data of occupied and vacant labels are sampled independently to ensure the same label imbalance. Each case is run ten times with different random number generator seeds. The topology, hyperparameters, and training strategies are kept the same. Figure 10a shows that the CNN-BiLSTM trained with a larger training dataset generally has a better occupancy detection performance. By training the model with a large epoch value with shuffled data order, it is able to compensate the small dataset by some extend.

Another important factor that impacts the model performance is label imbalance. In this study, we require that both labels should have more than 10% samples in the total data. This is because that the occupancy of an extremely imbalanced dataset is too easy to be detected. To verify this, sensitivity analysis of the CNN-BiLSTM performance to label imbalance is conducted by changing the ratio of vacant samples to the total samples. The case studies are repeated ten times with all the settings same as the previous sensitivity analysis. It is observed from Fig. 10b that by assigning the occupied condition to all the detection results (i.e., SPC=0 while SNS=1), the overall accuracy and F1 are close to 1. With a relatively more balanced dataset, the CNN-BiLSTM has satisfactory performance regarding all the metrics. Therefore, only the most challenging detection cases are included in the previous sections.

B. CNN-BiLSTM Applicability

A big concern of this research is how to extend the developed CNN-BiLSTM model from well-labelled data to real-world AMI data without occupancy labels. Generally, there are two appraches to establish occupancy labels for this large-scale application. The first approach is using thresholding [12] or unsupervised machine learning models [22]. The second approach is transfering the learned patterns between AMI and occupancy from models with accurate out-of-sample rate.

Another limitation of this research is that the dataset contains less diverse houses and limited socio-demographic information, although the ECO dataset is the largest dataset of its kind. Typically, commercial and industrial buildings have more regular occupancy patterns than residential buildings [22]. Therefore, we assume that the CNN-BiLSTM performance should at least have the same accuracy level when applied to occupancy detection of commercial and industrial buildings. Also, socio-demographic characteristics of the building, such as household age, house type, are expected to help the occupancy detection by grouping houses with similar patterns. However, this information is not available in the ECO dataset but will be explored by using other dataset or label discovery techniques [45] in the future.

VI. Conclusion

This paper developed a deep learning-based real-time building occupancy detection framework by using Advanced Metering Infrastructure (AMI) data. The sequentially stacked deep learning architecture performed latent feature learning through an end-to-end manner. The architecture consisted of a VGGNet-like four-layer convolutional neural network (CNN) and a three-layer bidirectional long short-term memory (BiLSTM) network. CNN learned spatial patterns from the AMI data and outputted feature maps. Then, the LSTM network derived forward and backward contextual dependencies from the CNN features. Last, a two-layer fully-connected dense layer and a sigmoid classification model generated the detection probability and was converted to the detection results by a threshold. Numerical experiments on a diverse publicly available dataset set showed that:

(1) The developed CNN-BiLSTM network accurately detected the building occupancy from purely AMI data with around **90%** accuracy;
(2) The CNN-BiLSTM model consistently and robustly outperformed the state-of-the-art classification models, including a k-Nearest Neighbours model, a support vector machine, a Gaussian process model , a random forest, a multi-layer perceptron classifier, and an adaptive boosting model;
(3) The developed CNN-BiLSTM model showed superior performance than other advanced deep learning architectures, including CNN, LSTM, CNN-LSTM without BiLSTM, and convolutional LSTM.
(4) The developed CNN-BiLSTM model showed high confidence in the detection and was less dependent on the classification threshold.

Future work will be conducted in four directions: (i) exploring deep learning with attention mechanisms in the occupancy detection, (ii) investigating the AMI-based occupancy prediction, (iii) detecting the occupancy of buildings without occupancy labels by transfer learning, and (iv) occupancy-enabled demand response & smart home management design.

Acknowledgement

We would like to thank the NVIDIA Corporation for donating the TITAN V GPU for this work.

REFERENCES

[1] S. Makonin, F. Popowich, I. V. Bajić, B. Gill, and L. Bartram, "Exploiting HMM sparsity to perform online real-time nonintrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2575–2585, Nov. 2016.

[2] C. Feng, M. Sun, and J. Zhang, "Reinforced deterministic and probabilistic load forecasting via *Q*-learning dynamic model selection," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1377–1386, Mar. 2020.

[3] C. Feng and J. Zhang, "Assessment of aggregation strategies for machine-learning based short-term load forecasting," *Electric Power Syst. Res.*, vol. 184, 2020, Art. no. 106304.

[4] G. Sun, Y. Cong, D. Hou, H. Fan, X. Xu, and H. Yu, "Joint household characteristic prediction via smart meter data," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1834–1844, Mar. 2019.

[5] N. Ahmed, M. Levorato, and G.-P. Li, "Residential consumer-centric demand side management," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4513–4524, Sep. 2018.

[6] J. Brooks, S. Kumar, S. Goyal, R. Subramany, and P. Barooah, "Energy-efficient control of under-actuated HVAC zones in commercial buildings," *Energy Build.*, vol. 93, pp. 160–168, Apr. 2015.

[7] M. Aftab, C. Chen, C.-K. Chau, and T. Rahwan, "Automatic HVAC control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system," *Energy Build.*, vol. 154, pp. 141–156, Nov. 2017.

[8] C. D. Korkas, S. Baldi, I. Michailidis, and E. B. Kosmatopoulos, "Occupancy-based demand response and thermal comfort optimization in microgrids with renewable energy sources and energy storage," *Appl. Energy*, vol. 163, pp. 93–104, Feb. 2016.

[9] A. Albert and R. Rajagopal, "Smart meter driven segmentation: What your consumption says about you," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4019–4030, Nov. 2013.

[10] J. Torriti, "Demand side management for the European supergrid: Occupancy variances of European single-person households," *Energy Policy*, vol. 44, pp. 199–206, May 2012.

[11] J. Chaney, E. H. Owens, and A. D. Peacock, "An evidence based approach to determining residential occupancy and its role in demand response management," *Energy Build.*, vol. 125, pp. 254–266, Aug. 2016.

[12] D. Chen, S. Barker, A. Subbaswamy, D. Irwin, and P. Shenoy, "Non-intrusive occupancy monitoring using smart meters," in *Proc. 5th ACM Workshop Embedded Syst. Energy Efficient Build.*, 2013, pp. 1–8.

[13] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models," *Energy Build.*, vol. 112, pp. 28–39, Jan. 2016.

[14] J. Zou, Q. Zhao, W. Yang, and F. Wang, "Occupancy detection in the office by analyzing surveillance videos and its application to building energy conservation," *Energy Build.*, vol. 152, pp. 385–398, Oct. 2017.

[15] H. Zou, Y. Zhou, J. Yang, and C. J. Spanos, "Device-free occupancy detection and crowd counting in smart buildings with WiFi-enabled IoT," *Energy Build.*, vol. 174, pp. 309–322, Sep. 2018.

[16] Z. Chen, Q. Zhu, and Y. C. Soh, "Smartphone inertial sensor-based indoor localization and tracking with iBeacon corrections," *IEEE Trans. Ind. Informat.*, vol. 12, no. 4, pp. 1540–1549, Aug. 2016.

[17] N. Li, G. Calis, and B. Becerik-Gerber, "Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations," *Autom. Constr.*, vol. 24, pp. 89–99, Jul. 2012.

[18] W. Wang, T. Hong, N. Li, R. Q. Wang, and J. Chen, "Linking energy-cyber-physical systems with occupancy prediction and interpretation through WiFi probe-based ensemble classification," *Appl. energy*, vol. 236, pp. 55–69, Feb. 2019.

[19] R. Razavi, A. Gharipour, M. Fleury, and I. J. Akpan, "Occupancy detection of residential buildings using smart meter data: A large-scale study," *Energy Build.*, vol. 183, pp. 195–208, Jan. 2019.

[20] J. G. Ortega, L. Han, N. Whittacker, and N. Bowring, "A machine-learning based approach to model user occupancy and activity patterns for energy saving in buildings," in *Proc. Sci. Inf. Conf. (SAI)*, London, U.K., 2015, pp. 474–482.

[21] J. Zhao, B. Lasternas, K. P. Lam, R. Yun, and V. Loftness, "Occupant behavior and schedule modeling for building energy simulation through office appliance power consumption data mining," *Energy Build.*, vol. 82, pp. 341–355, Oct. 2014.

[22] M. Jin, R. Jia, and C. J. Spanos, "Virtual occupancy sensing: Using smart meters to indicate your presence," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3264–3277, Nov. 2017.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[24] H. Zou, Y. Zhou, J. Yang, and C. J. Spanos, "Towards occupant activity driven smart buildings via WiFi-enabled IoT devices and deep learning," *Energy Build.*, vol. 177, pp. 12–22, Oct. 2018.

[25] Y. Wang, Q. Chen, D. Gan, J. Yang, D. S. Kirschen, and C. Kang, "Deep learning-based socio-demographic information identification from smart meter data," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2593–2602, May 2019.

[26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," 2015. [Online]. Available: arXiv:1511.07289.

[27] J.-F. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève, "Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1203–1215, Mar. 2019.

[28] G. Dai, J. Xie, and Y. Fang, "Siamese cnn-bilstm architecture for 3D shape representation learning," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 670–676.

[29] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," 2018. [Online]. Available: arXiv:1801.02143.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.

[31] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Vancouver, BC, Canada, 2013, pp. 6645–6649.

[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[33] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, "The ECO data set and the performance of non-intrusive load monitoring algorithms," in *Proc. 1st ACM Conf. Embedded Syst. Energy Efficient Build.*, 2014, pp. 80–89.

[34] W. Kleiminger, C. Beckel, and S. Santini, "Household occupancy monitoring using electricity meters," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2015, pp. 975–986.

[35] W. Kleiminger, C. Beckel, T. Staake, and S. Santini, "Occupancy detection from electricity consumption data," in *Proc. 5th ACM Workshop Embedded Syst. Energy Efficient Build.*, 2013, pp. 1–8.

[36] B. Dong, K. P. Lam, and C. Neuman, "Integrated building control based on occupant behavior pattern detection and local weather forecasting," in *Proc. 12th Int. IBPSA Conf. Sydney IBPSA Aust.*, 2011, pp. 14–17.

[37] L. Yang, K. Ting, and M. B. Srivastava, "Inferring occupancy from opportunistically available sensor data," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Budapest, Hungary, 2014, pp. 60–68.

[38] G. Diraco, A. Leone, and P. Siciliano, "People occupancy detection and profiling with 3D depth sensors for building energy management," *Energy Build.*, vol. 92, pp. 246–266, Apr. 2015.

[39] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[40] C. Feng, M. Cui, B.-M. Hodge, S. Lu, H. Hamann, and J. Zhang, "Unsupervised clustering-based short-term solar forecasting," *IEEE Trans. Sustain. Energy*, vol. 10, no. 4, pp. 2174–2185, Oct. 2019.

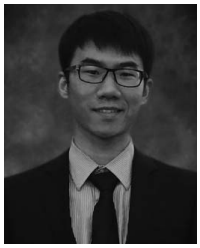
[41] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *Int. J. Data Mining Knowl. Manag. Process*, vol. 5, no. 2, p. 1, 2015.

[42] M. Gönen, *Analyzing Receiver Operating Characteristic Curves with SAS*. Cary, NC, USA: SAS Inst., 2007.

[43] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[44] T. Hong, P. Pinson, and S. Fan, "Global energy forecasting competition 2012," *Int. J. Forecasting*, vol. 30, no. 2, pp. 357–363, 2014.

[45] Y. Cong, G. Sun, J. Liu, H. Yu, and J. Luo, "User attribute discovery with missing labels," *Pattern Recognit.*, vol. 73, pp. 33–46, Jan. 2018.



Cong Feng (Student Member, IEEE) received the B.S. degree in power engineering from Wuhan University, Wuhan, China, in 2014, and the M.S. degree in mechanical engineering from the University of Texas at Dallas, Richardson, TX, USA, in 2017, where he is currently pursuing the Ph.D. degree with the Department of Mechanical Engineering.

His research interests include machine/deep learning, power system big data analytics, and power system time series forecasting.

Ali Mehmani (Member, IEEE) received the B.Sc. degree in mechanical engineering from University of Tehran, Tehran, Iran, in 2007, the M.Sc. degree in aerospace engineering from K. N. Toosi University of Technology, Tehran, Iran, in 2010, and the Ph.D. degree in mechanical engineering from Syracuse University, Syracuse, NY, USA, in 2015.

From 2015 to 2018, he was a Postdoctoral Research Associate with Data Science Institute, Columbia University, New York, NY, USA. He is currently the Head of the Data Science and Analytics, Prescriptive Data, New York, and an Adjunct Research Scientist with the Data Science Institute, Columbia University. He specializes in multidisciplinary optimization, complex system design, machine learning, deep structural networks, and cyber-physical systems, with primary applications in building energy-efficiency.

Jie Zhang (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively, and the Ph.D. degree in mechanical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2012.

He is currently an Assistant Professor with the Department of Mechanical Engineering, University of Texas at Dallas. His research interests include multidisciplinary design optimization, complex engineered systems, big data analytics, wind and solar forecasting, renewable integration, and energy systems modeling and simulation.