Journal of Computers Vol. 31 No. 5, 2020, pp. 159-175
doi:10.3966/199115992020103105012

Smart Occupancy Detection System Based on Long Short-Term Memory Units

Asif Husnain, Tae-Young Choe*

Department of Computer Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, Korea
ahsnyn@gmail.com, choety@kumoh.ac.kr

Received 2 May 2019; Revised 1 October 2019; Accepted 21 November 2019

Abstract. Smart lighting is a system designed to consume lighting energy efficiently. Occupancy detection is one of the key functionalities for a smart lighting system or home automation. The previous researches expect that room occupancy can be monitored by some sensors, i.e. remote thermal sensor arrays and pyroelectric sensors. Unfortunately, they cannot detect occupancy of the entire room with the limited number of sensors. In order to detect occupancy even in the off-range area of thermal sensors, we proposed a deep learning based occupant detection system comprising a 4×4 thermal sensor array and a PIR sensor. The proposed system is focuses on the occupancy detection of the whole room instead of the occupancy detection in front of the sensor area only. The deep learning module consists of Long Short-Term Memory (LSTM) units in order to achieve robust occupancy detection. The proposed system can memorize the sequence of human movements and detects occupancy of the room with high accuracy. The performance of the proposed system is compared with several state-of-the-art machine learning techniques and achieves 95.62% accuracy on test data set.

Keywords: deep learning, lighting control, LSTM, occupancy detection, sensor system

1 Introduction

The United States of America's Energy Information Administration (EIA) evaluated that around 273 billion kilowatt-hours (kWh) was utilized for lighting competence in residential and commercial facilities in 2017. It was equivalent to approximately 10% of entire USA electricity consumption [1]. In home and offices, rooms are lighted up assuming maximum occupancy, which leads to lighting empty rooms and results in loss of electrical energy. As green energy preservation initiatives become dominant, some initiatives should be taken to trim light usage from empty rooms and to save electricity consumption.

Occupancy detection is the problem of detecting existence of an individual in a specified area. Occupancy detection is one of the key functionalities of smart lighting and ventilation systems with an important role of energy saving. Robust occupancy detection can assist to decrease electricity consumption and is important to improve the safety and security of the residents by aiding emergency evacuations, observing security, and rescue procedures. As buildings turn out to be smart, they are primarily dependent on sensors to control numerous tasks and energy consumption.

The major attentions in designing an occupancy detection system, are considering low-cost, reliable, and privacy conserving system. Such requirements force the occupancy detection systems to use simple and low-priced sensor devices instead of sophisticated and expensive devices like a wide angle camera with fisheye lens. In order to make a reliable occupancy detection system, sensor off-range area where any sensor cannot observe something should be minimized. One method of minimizing the sensor off-range areas using the simple sensors is to increase the number of sensors to cover the entire room. But it would be a burden to the occupancy detection system in terms of installation cost. Another method is to select suitable types of sensor devices, to locate the sensors in some good positions, and to use a decision algorithm that can detect occupancy in the sensor off-range area.

* Corresponding Author

Many neural networks of previous works are used in order to learn and detect current sensor data patterns, which cannot detect or predict movement in the sensor off-range area. For example, consider a monitor room where an operator uses three consoles M1, M2, and M3 where console M3 locates out of sensor ranges. Assume that the operator moves in the following two patterns: he / she gets off work after finishing a reporting operation at M1; he / she uses M1, M2, and M3 in sequence during a short time and returns to M1 again for general management. A simple occupancy detection algorithm would mistake to decide such that no human occupies the room in the middle of the second case. Deep neural networks like RNN are good candidates for predicting human movement patterns based on previous movement patterns [2]. Recently, Long Short-Term Memory (LSTM) shows superior performance among RNNs [3].

In this paper, we propose an occupancy detection system which consists of a LSTM decider module, an Omron D6T 4×4 thermal sensor array [4], and a pyroelectric (PIR) sensor [5] for cost effective high accuracy. The thermal sensor array measures temperature in a 4×4 grid pattern within the 3×3 meter (m) area range. Thermal sensor array minimizes disclosure of the inside information, thus keeps privacy, but can provide suitable amount of sensor data to the decision algorithm. The PIR sensor is installed near the room door to detect human entrance and exit. Two sensors work well during nighttime because they use infrared modules. Long Short-Term Memory (LSTM) units are used to decide human occupancy given the sensor data. LSTM shows good performance when sequential patterns should be predicted. One problem of decision making using LSTM is that lots of hyper parameters should be examined and adjusted in order to get high performance. The hyper parameters are deeply examined in order to find their properties in sequential data of occupancy detection sensors.

The main contributions of this paper are listed as follows:

- An occupancy detection system can be constructed without fully covering the entire room using sensors. A decision module using LSTM covers the sensor off-range area by predicting human movement pattern. It reduces installation cost of occupancy detection system.
- Fine tuning on LSTM hyper parameters reduces the number of required neuron nodes. A LSTM with smaller number of nodes can show higher accuracy.
- Even small number of simple sensors can detect human occupancy with high accuracy while privacy is preserved.

The remainder of this paper is organized as follows: In Section 2, we discuss related works. We introduce the proposed occupancy detection system in Section 3, and present experimental results in Section 4. We conclude the paper in Section 5.

2 Related Works

Chen et al. suggested a classification of occupancy detection / estimation systems based on sensor types, which include PIR (Passive InfraRed), smart meter, environmental sensors like CO2, temperature, humidity, light, pressure sensor, camera, WiFi, BLE, RFID, and sensor fusion [6]. Each sensor has its advantages and disadvantages. Lu et al. and Agarwal et al. proposed occupancy detection systems using PIR sensors [9-10]. Installing a PIR sensor to the door reduces the accuracy of occupancy because a PIR sensor cannot exactly count the number of people who pass the door together. As the result, the systems are unable to correctly count the number of people inside the room. Scott et al. used RFID tags to estimate occupancy in a room [9]. Qolomany et al. proposed an occupancy prediction using WiFi detection and LSTM. [10]. Although the RFID tags can be used to estimate occupancy exactly, each RFID tag must be carried by every individual. BLE and smartphone with WiFi module have the same shortcoming. Lam et al. and Mamidi et al. used environmental sensors, i.e. carbon dioxide, carbon monoxide, lighting, temperature, humidity, motion, and acoustics [11-12]. Their systems construct feature vectors from the multiple sensors and applies them to several neural network models to detect occupancy. This approach does not work well when it is combined with a ventilation strategy. Ventilation changes carbon dioxide and humidity level, which affects the accuracy of occupancy detection. Amin et al. proposed an occupancy detection system that uses a visual camera and a thermal sensor array for detecting occupancy with a neural network classifier [13]. Main problems of using camera are privacy and computing power.

Another classification of occupancy detection / estimation system is done by algorithms or Artificial Intelligence (AI) modules that decide occupancy in an area. The AI modules can be classified to two groups: general AI algorithms and deep neural network. The general AI algorithms include Logistic Regression 2-class classifier (LR) [14], Decision Trees classifier (C4.5) [15], Random Forest classifier (RF) [16], K-Neighbors classifier (KNN) [17], Multilayer Perceptron (MLP) [18], Support Vector classifier (SVC) [19], XG Boost classifier (XGB) [20], Cat Boost classifier (CB) [21], and Light GBM classifier (LGBM) [22]. Occupancy detection methods that use the general AI algorithms are suitable when the computing power is low.

Deep neural networks include RNN and LSTM [3]. Deep neural network shows higher accuracy with the cost of high computing power. Qolomany et al.'s occupancy detection method uses one layer LSTM and shows about 6.4%~7.3% improvement on root mean square error (RMSE) compared to Auto Regression Integrating Moving Average (ARIMA) method [10]. Chen et al. used a convolutional deep bidirectional LSTM with multi-type sensors for better estimation including sensor off-range areas [23]. Although the detection accuracy is improved about 5% compared to the previous works, large computation module which includes 4 LSTM layers and a convolutional network is another power consumption problem.

Since accurate data from sensors is the major component for good occupancy detection, location and degree of sensors have been considered in order to minimize sensor off-range areas. ThermoSense system was experienced with a ceiling mount setting using Panasonic Grid-Eye and a PIR sensor [24]. The PIR sensor achieves a simple motion detection in a specific area and the Grid-Eye thermal array is used to detect occupancy where motion is not detected. The problem of ThermoSense system is the limited cover area of the room. Ash et al. mounted a thermal sensor array at ceiling of 2.6m height and oriented the direction of the sensor to 30 degree from wall in order to maximize occupant detection area in the thermal sensor array [25]. A 4×16 thermal sensor array and a PIR sensor are used and multiple decision algorithms are tested. Among them, C4.5 decision tress showed the highest accuracy.

Occupancy detection systems have suffered from the sensor off-range areas except some environment sensors like CO_2, temperature, and humidity sensors. Chen et al. used various sensors like environment sensors and pressure sensors in an office room [23]. Since pressure sensors indicate occupancy directly, errors occur when people move around areas where the pressure sensors are not installed. Husnain and Choe proposed an occupancy detection system that tries to detect occupancy using well-located sensors and human pattern prediction even when people is not detected by any sensor [26]. In order to predict human behavior pattern, a 3-layered LSTM with 535,960 neuron nodes are used. A thermal sensor array is mounted above a bookshelf and monitors 1/3 of a room at an angle of 70 degree from the bookshelf. A PIR sensor is located near the door of the room in order to check human entrance and exit. Although the system shows 85% accuracy on their experimental environment, the neural network is heavy and is not tuned well.

3 Proposed Scheme

We propose a robust occupancy detection system "Smart Occupancy Detection" (SOD) based on LSTM units with a 4×4 thermal sensor array and a PIR sensor. Fig. 2 shows the structure of SOD system. Both sensors are interfaced to a microcontroller Arduino UNO using Arduino integrated development environment [27], which sends the sensor data to a Raspberry Pi 3 [28]. The Py-serial library [29] is used to get sensor's output through a python script [30]. The LSTM module classifies the input as occupied in the form of 1 or not occupied in the form of 0. Fig. 2 shows the structure of the proposed system.

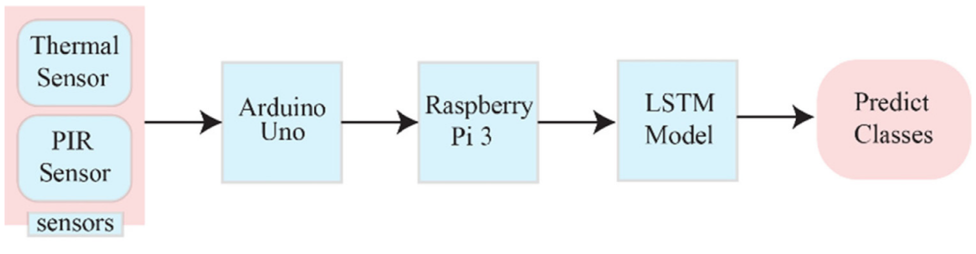


Fig. 2. Structure of the SOD

3.1 Dataset and Pre-processing

In order to gather training and test dataset with different human behaviors, an experimenter moves in diverse areas including in-range and off-range of thermal sensor array regions. The dataset is collected according to four human behavior categories. All categories are observed in 1m, 2m and 3m distances from the thermal sensor. The experimenter frequently moves between off-range and in-range sensing regions. The experimenter stays in the off-range region during 1 to 3 minutes before returning to in-range regions. In the behavior category 1 (C1), the experimenter moves within 1m, 2m and 3m in-range regions. In the behavior category 2 (C2), the experimenter moves within 1m, 2m and 3m in-range regions while other experimenter goes out and comes inside the room. In the behavior category 3 (C3), the experimenter moves within 1m, 2m and 3m in-range regions and leaves / enters the room sometimes. In the behavior category 4 (C4), the experimenter moves within 1m, 2m and 3m in-range regions, moves to off-range regions, and returns to in-rage regions. Fig. 3 illustrates the overview of the experiment room with the movement regions and the locations of the sensors. The thermal sensor array is installed in 1.8m height of the bookshelf with 70 degree from the bookshelf. The PIR sensor is installed near the door and used to detect human entrance and exit.

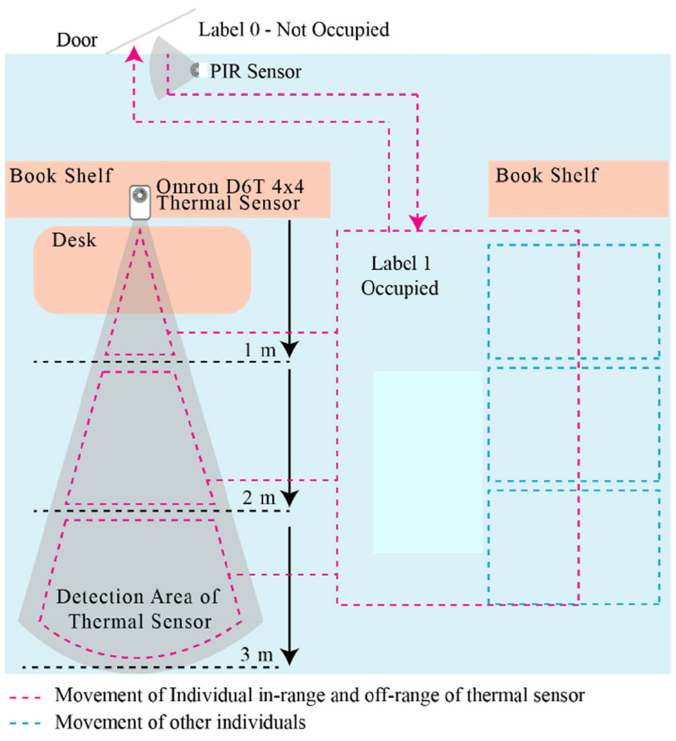

Door
Label 0 - Not Occupied
PIR Sensor
Book Shelf
Omron D6T 4x4 Thermal Sensor
Book Shelf
Desk
Label 1 Occupied
1 m
2 m
3 m
Detection Area of Thermal Sensor
Movement of Individual in-range and off-range of thermal sensor
Movement of other individuals

Fig. 3. Overview of the room

A data collection system is constructed as shown in Fig. 4. The collected sensor data is stored in Apache Cassandra Database [31] server along with timestamp while experimenters move around the room according to a scenario. The scenario indicates times that the experimenter enters in-range areas, moves to off-range areas, goes out of the room, and return to in-range regions of the room. After executing the scenario, data stored in the database is retrieved back to comma-separated Values (CSV) files with timestamps. The experimenter writes down labels into the CSV files according to the action time in the scenario and timestamps such as 1 if he / she is in the room, 0 otherwise. Timestamp column is removed when the data is used for training and testing, because timestamp can be a redundant hint to the LSTM network. Thus, there are only 17 features (a PIR sensor value and 16 thermal sensor array values) remaining. The total dataset includes 24,840 samples and each sample row consists of 17 features and an occupancy label. Dataset of each category is split into 60% (14904 samples) of training set, 20% (4968 samples) of validation set and 20% of test set. The network was trained using the training set, while the test set was used for unseen prediction to get model performance.

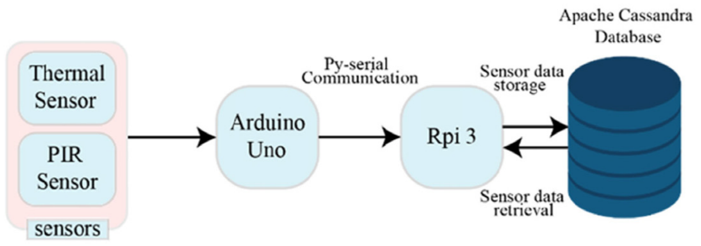


Fig. 4. Collecting sensor data into database for learning

3.2 Proposed SOD System Model

The SOD system is designed considering two points: lightweight deep neural network and optimized hyper parameters. The system consists of two LSTM layers and a dense layer in order to reduce execution time (Fig. 5). Each hyper parameter is experimented on several values to achieve the higher accuracies. A data in a time consists of 17 features from two sensors, a PIR and a thermal sensor array. A sequential contiguous data is an input to LSTM layer 1. The size of the contiguous data, or time step is a hyper parameter which ranges from 1 to 50. The number of LSTM units in layer 1 is another hyper parameter and ranges from 100 to 300. A ReLU (Rectified Linear Unit) activation function is applied to each LSTM layer 1 output. The output of the activation functions is provided to LSTM layer 2. The output of LSTM layer 2 is applied to a dense layer which has dropout rate 0.2. Two neurons are used in a dense layer (D1) along with sigmoid activation function. Two neurons in the final output layer are triggered (1, 0) for occupancy and (0, 1) otherwise.

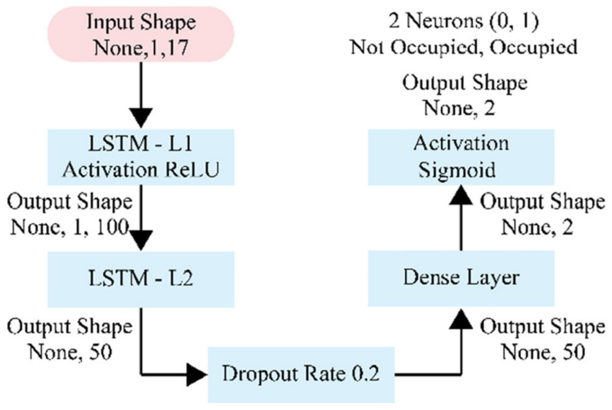


Fig. 5. SOD Model

Table 1 shows the output shape of each layer along with some fixed hyper parameters where time step is 1, the number of units in LSTM layer 1 is 100, the number of units in LSTM layer 2 is 50.

Table 1. Output shape of each layer with base parameters

Layer (type)	Output Shape	Parameters
LSTM – L1 (with ReLU)	(None, 100)	81,200
LSTM – L2	(None, 50)	30,200
Dropout Rate (0.2)	(None, 50)	0
Dense Layer	(None, 2)	102
Sigmoid Activation	(None, 2)	0
Total parameters:		111,502

4 Experimental Results

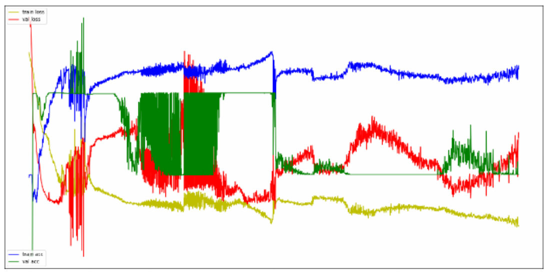
The proposed system SOD is implemented using Keras library [21] with TensorFlow [22] as a backend. Sequential model in Keras is used to implement stacked LSTM. SOD runs in a Linux computer with a core-i7 CPU, a GeForce GTX1080 GPU, and 16GB memory.

4.1 Hyper-parameter Selection

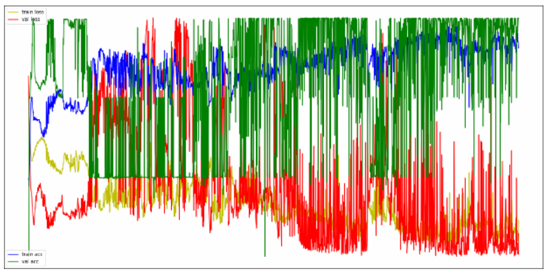
There are some hyper-parameters which affect the performance of SOD: batch size, epoch number, the number of LSTM units per layer, and time step. Time step means the number of contiguous inputs to LSTM network. For example, time step 1 means only the current data is used as an input; time step 5 means the previous 4 data and the current data are used as an input. In general, sequential data $\boldsymbol{d}_{t-s+1}, \boldsymbol{d}_{t-s+2}, \cdots, \boldsymbol{d}_t$ are used as an input if current time is t, data in time t is d_t, and s is the time step.

In order to decide the batch size first, we set time step as 1 and applied batch sizes, i.e. 32, 64, 128, 256, 512, 1024, 1536, and 2048. The epoch number is selected as 2000 to observe training accuracy and loss trends along with validation accuracy and loss. In the figures from Fig. 6 to Fig. 15, the x-axis is the number of epochs ranged from 1 to 2000, left y-axis indicates training and validation losses ranged from 0.00 to 0.30, while the right y-axis indicates training and validation accuracies ranged from 0.00 to 1.00. In the figures, blue color curves represent training accuracies, while green color curves represent validation accuracies. Training and validation losses appears in yellow and red curves, respectively.

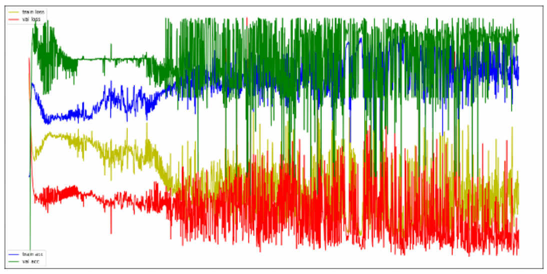
Fig. 6 and Fig. 7 show accuracies and losses trends of various batch sizes when the time steps are 1 and 2, respectively. After carefully observing the cases of time step 1 with different batch sizes, Fig. 6(g) of batch size 1536 and Fig. 6(h) of batch size 2048 shows stable accuracies and losses trends. As the training and validation loss trends are stable, we can see that over-fitting or under-fitting do not happen. Similarly, from Fig. 7(f) and Fig. 7(g), the batch size of 1024 and 1536 for time step 2 shows the stable accuracies and losses trends. Table 2 shows the training and validation accuracies of hyper-parameters where training and validation loss trends are stable. After observing the results from time step 1 and 2, we find that small batch sizes under 1024 make the network unstable. Thus, the small batch sizes are not considered in further experiments. For time step 1, Fig. 6 shows that loss and accuracy curves are stable only at batch size 1536 and 2048, but it is not sure whether it is the same for other time steps or not. Fig. 8 shows that batch size 2048 is not suitable when time step is 2, which means that multiple hyper-parameters should be considered together instead of deciding single hyper-parameter one by one.



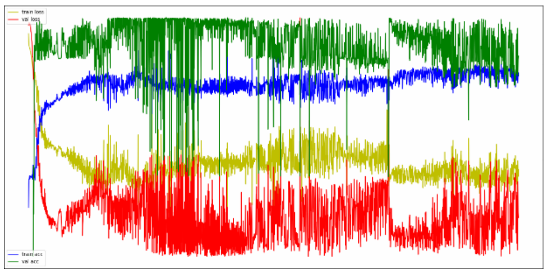
(a) batch size 32



(b) batch size 64

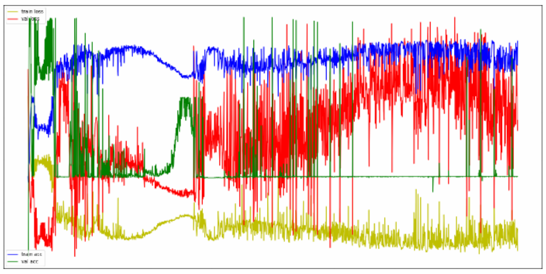


(c) batch size 128

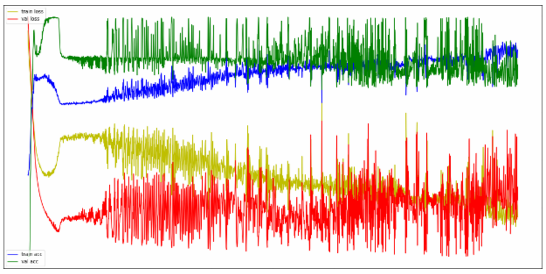


(d) batch size 256

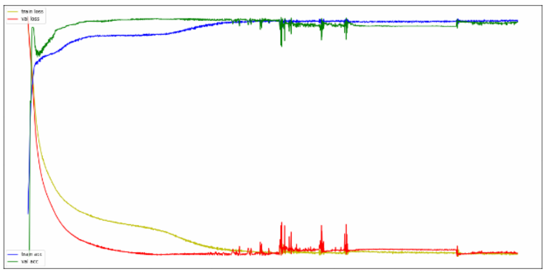
Fig. 6. Accuracies and losses in the case of time step 1



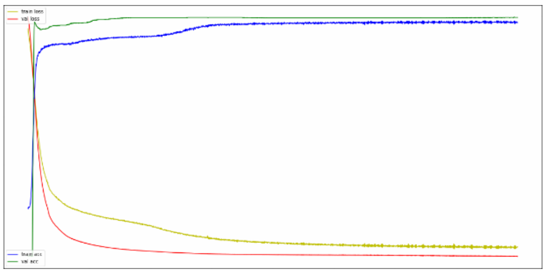
(e) batch size 512



(f) batch size 1024



(g) batch size 1536

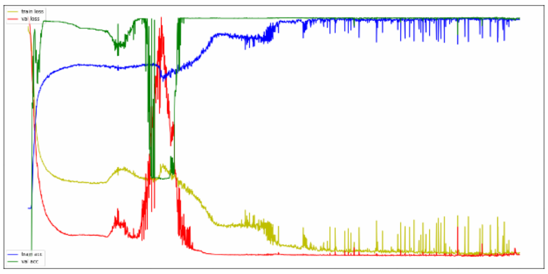


(h) batch size 2048

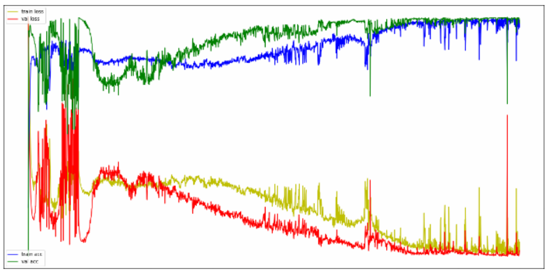
Fig. 6. Accuracies and losses in the case of time step 1 (continue)

Table 2. Training and validation accuracies of time step 1 and step 2 with the epoch of 2000

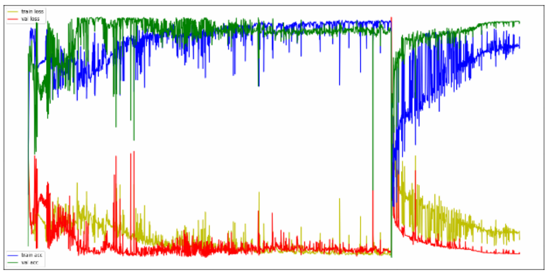
Time step	Batch size	Training Accuracy (%)	Validation Accuracy (%)
1	1536	97.7	97.35
1	2048	97.02	95.4
2	1024	98.7	96.7
2	1536	96.9	96.12



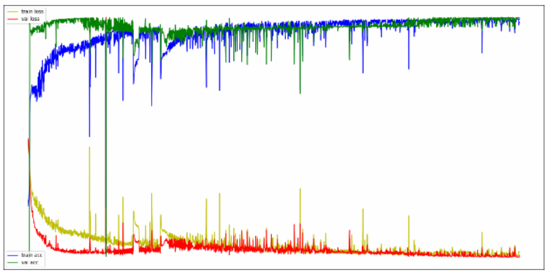
(a) batch size 32



(b) batch size 64

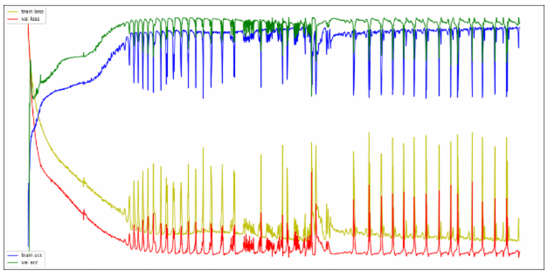


(c) batch size 128

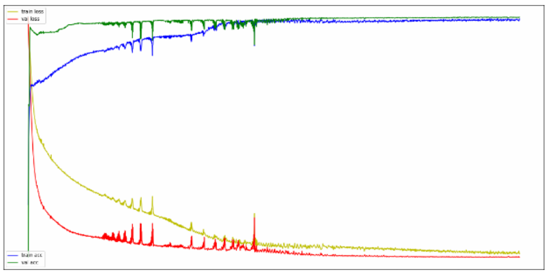


(d) batch size 256

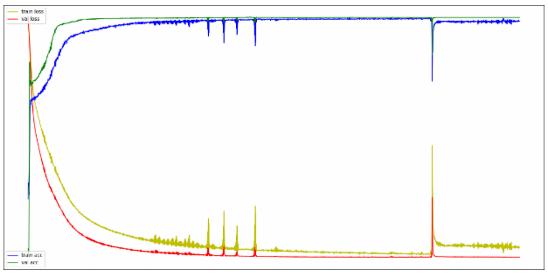
Fig. 7. Accuracies and losses in the case of time step 2



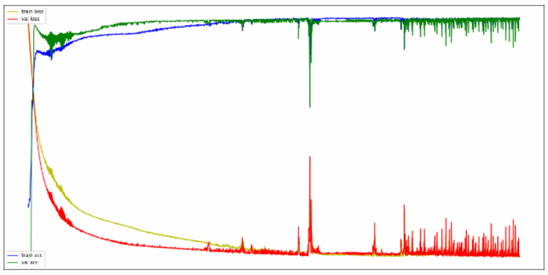
(e) batch size 512



(f) batch size 1024

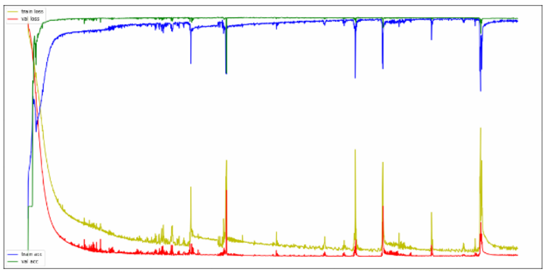


(g) batch size 1536

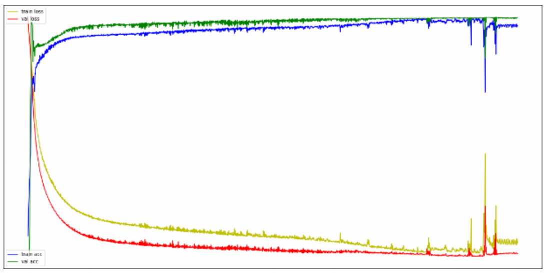


(h) batch size 2048

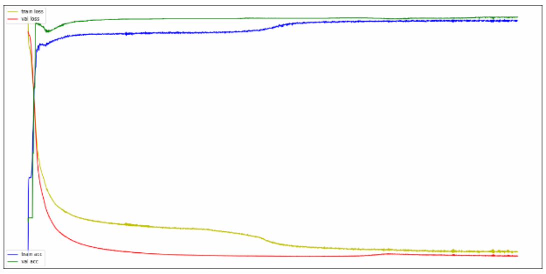
Fig. 7. Accuracies and losses in the case of time step 2 (continue)



(a) batch size 1024



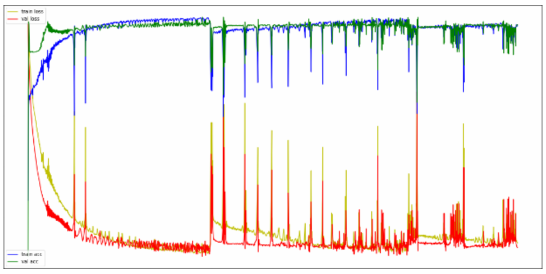
(b) batch size 1536



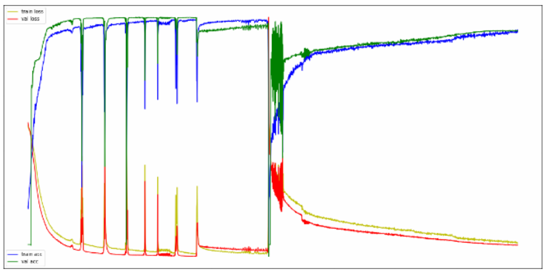
(c) batch size 2048

Fig. 8. Accuracies and losses in the case of time step 3

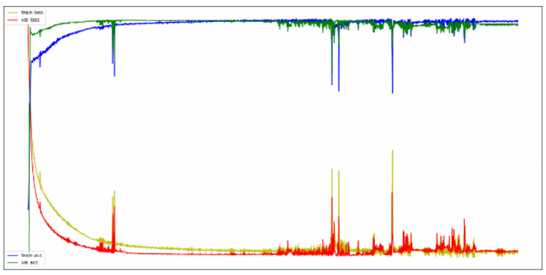
In order to find properties of time step, time steps 3, 4, and 5 are applied with batch size 1024, 1536, and 2048. Fig. 9, Fig. 10, and Fig. 11 show accuracy and loss trends of various batch sizes when previous time steps are 3, 4 and 5, respectively.



(a) batch size 1024



(b) batch size 1536



(c) batch size 2048

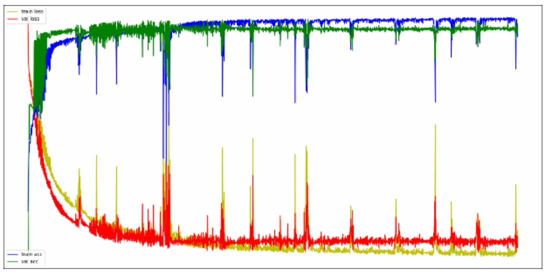
Fig. 10. Accuracies and losses in the case of time step 5

After carefully observing time step 3 with batch sizes of 1024, 1536 and 2048, Fig. 8(c) of batch size 2048 shows stable accuracies and losses trends. Also, batch size 2048 shows stable trends in the case of time step 4 and 5 as shown in Fig. 9(c) and Fig. 10(c). Table 3 summarizes the training and validation accuracies when batch size is 2048. The batch size of 2048 is used for further experiments on time steps 10, 20, 30, 40 and 50.

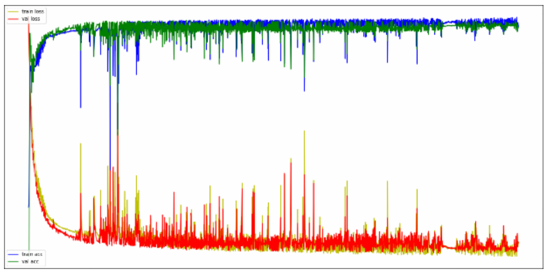
Table 3. Training and validation accuracies of time step 3, 4 and 5

Time step	Batch size	Training Accuracy (%)	Validation Accuracy (%)
3	2048	97.4	95.6
4	2048	97.4	96.80
5	2048	97.08	95.01

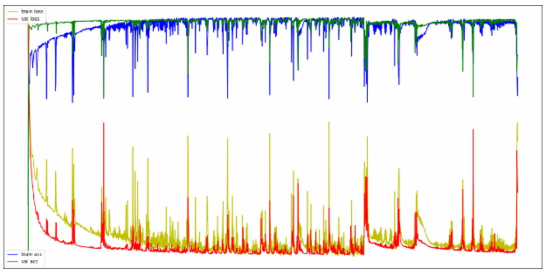
Experiments on time step from 1 to 5 show that SOD is quite sensitive to hyper-parameters. We tried to observe if larger time steps can stabilize the LSTM network. In the case, more LSTM units should be considered in order to remember input patterns. Thus, effects of longer time steps and larger number of LSTM units are examined. Layer 1 that has 100 LSTM units is denoted as L1-100 and layer 2 that has 50 LSTM units is denoted as L2-50. Fig. 11, Fig. 12, Fig. 13, Fig. 14 and Fig. 15 shows accuracies and losses trend of previous time steps 10, 20, 30, 40 and 50, respectively. With batch size 2048 and epoch size 2000.



(a) L1-100 and L2-50

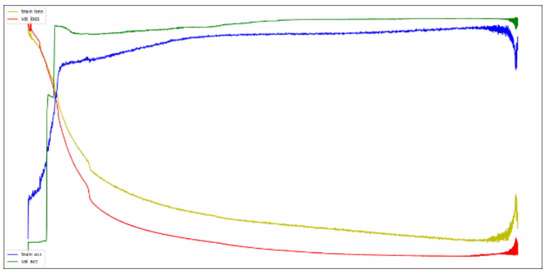


(b) L1-200 and L2-100

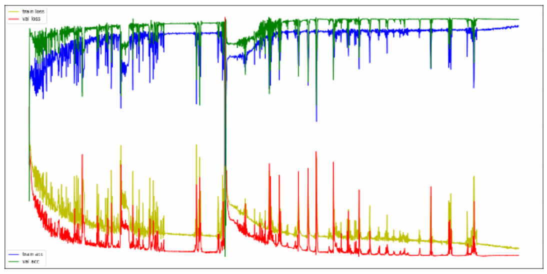


(c) L1-300 and L2-150

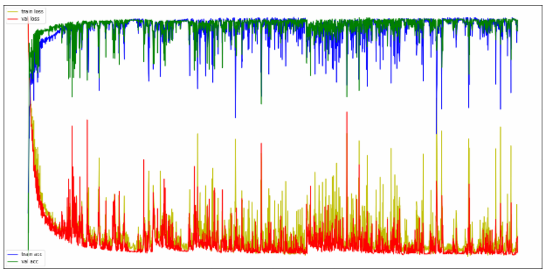
Fig. 11. Accuracy and loss trends of SOD for time step 10 and batch size 2048



(a) L1-100 and L2-50

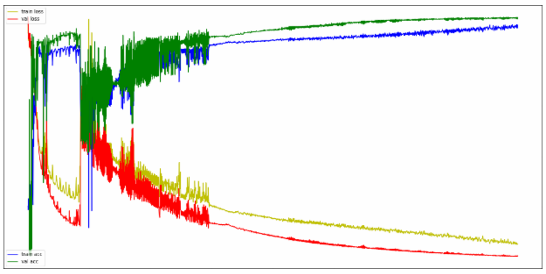


(b) L1-200 and L2-100

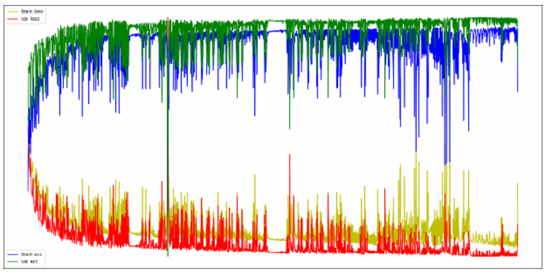


(c) L1-300 and L2-150

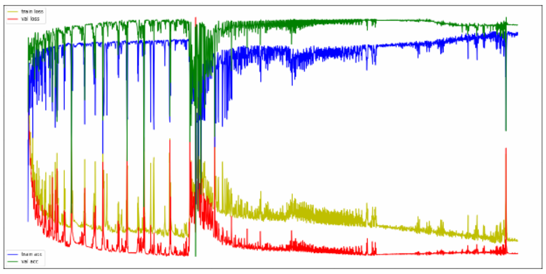
Fig. 12. Accuracy and loss trends of SOD for time step 20 and batch size 2048



(a) L1-100 and L2-50

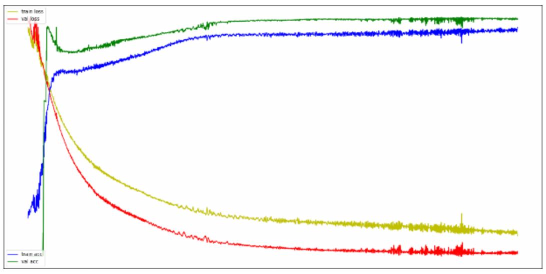


(b) L1-200 and L2-100

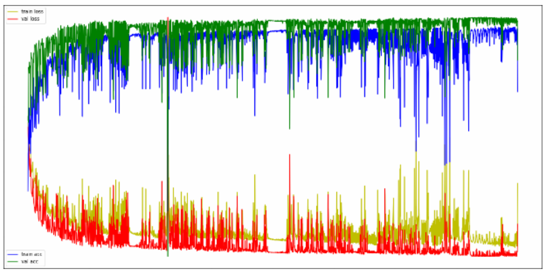


(c) L1-300 and L2-150

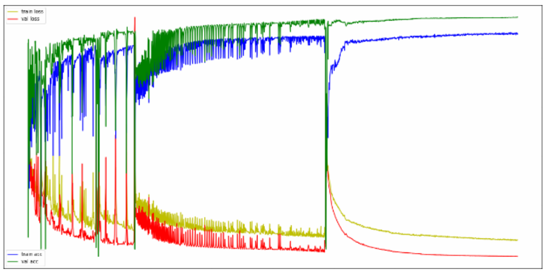
Fig. 13. Accuracy and loss trends of SOD for time step 30 and batch size 2048



(a) L1-100 and L2-50



(b) L1-200 and L2-100



(c) L1-300 and L2-150

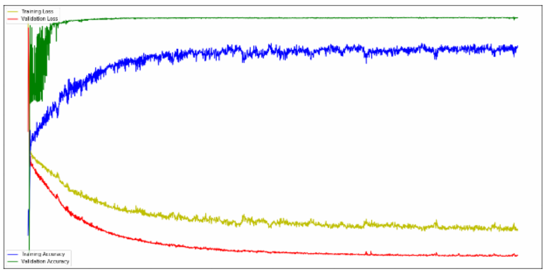
Fig. 14. Accuracy and loss trends of SOD for time step 40 and batch size 2048

According to Fig. 11 to Fig. 15, the SOD of L1-100 and L2-50 shows stable accuracy and loss except time step 10. Other layer constructions show unstable trends except L1-300 and L2-150 with time step 10, which makes it hard to decide some suitable hyper-parameters in the range. From Fig. 6 to Fig. 15, we observed that smaller time steps show stable performance compared to larger time steps. Although the configuration with batch size 1024 and time step 2 shows the highest accuracy, its nearby configurations do not show stable trends. Thus, the configuration with batch size 2048, epoch size 2000, and time step 4 shown in Table 3 is selected for SOD.

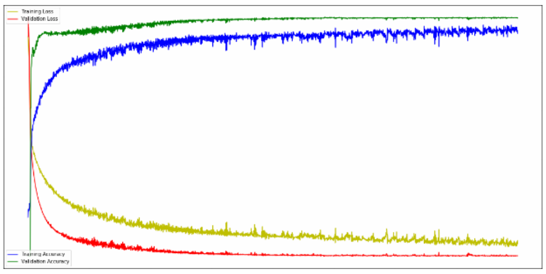
4.2 Effect of LSTM Layers and Nodes

After selecting the efficient configuration of hyper-parameters including the number of epochs, batch size, and time steps, we decided the remaining hyper parameters: the number of LSTM layers and the number of LSTM nodes. We collected candidate configurations which show stable trends as epoch increases, because neural network with unstable accuracy learning trend will show unpredictable accuracy.

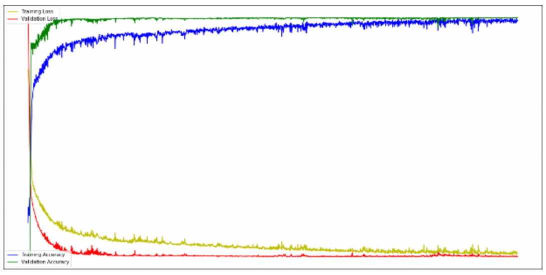
Fig. 16 shows accuracies and losses of SOD with only one LSTM layer having 100, 200, and 300 LSTM units, respectively. Fig. 17 shows results of SOD learning with two LSTM layers: (L1-100, L2-50), (L1-200, L2-100), and (L1-300, L2-150), respectively. Fig. 18 shows results of SOD learning with three layered stacked LSTM model: (L1-100, L2-75, and L3-50), (L1-200, L2-150, and L3-100), and (L1-300, L2-225, and L3-150), respectively. The graphs have epoch size 2000.



(a) L1-100

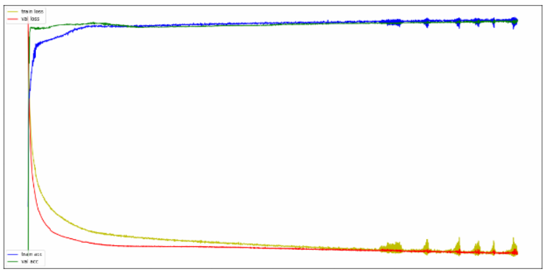


(b) L1-200

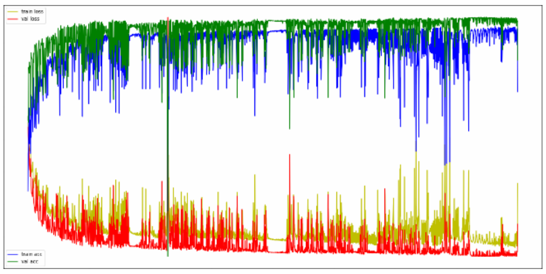


(c) L1-300

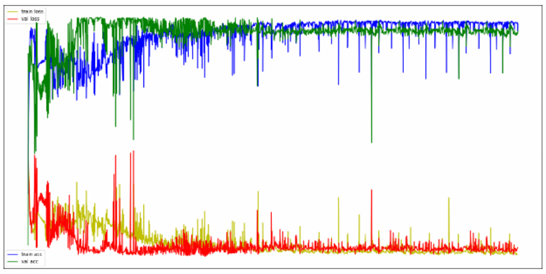
Fig. 16. Accuracy and loss trends of SOD with one LSTM layer only



(a) L1-100 and L2-50

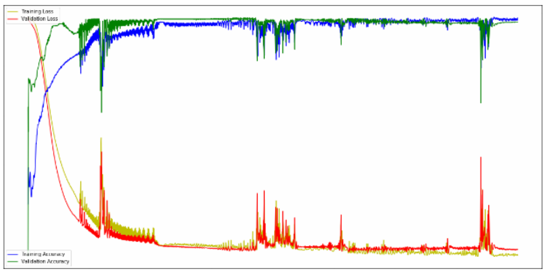


(b) L1-200 and L2-100

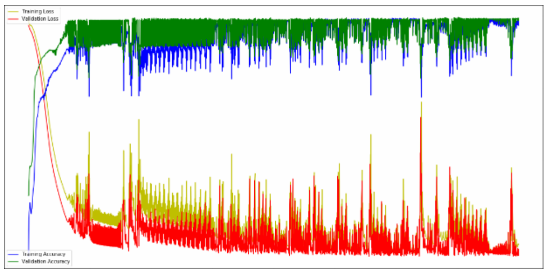


(c) L1-300 and L2-150

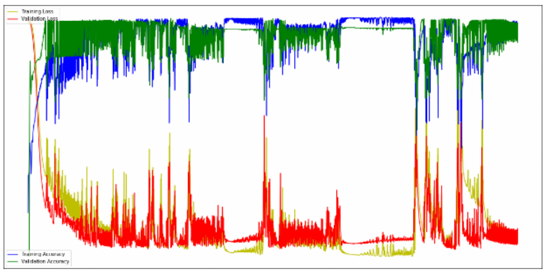
Fig. 17. Accuracy and loss trends of SOD with two stacked LSTM layers



(a) L1-100, L2-75 and L3-50



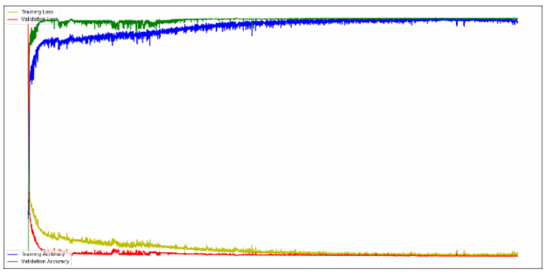
(b) L1-200, L2-150 and L3-100



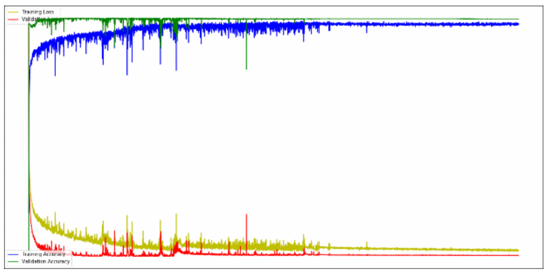
(c) L1-300, L2-225 and L3-150

Fig. 18. Accuracy and loss trends of SOD with three stacked LSTM layers

According to the figures, the one-layer LSTM with L1-300 and the two stacked LSTM with (L1-100, L2-50) show stable accuracy and loss trends. Thus, the two configurations are included in our candidate configurations. Although the one-layer LSTM with L1-100 shows stable trends (Fig. 16(a)), it is assumed to be under-fitted because validation accuracy is higher than training accuracy. Since the accuracy of L1-100 does not show increasing trend after epoch 400, it is considered to have too small LSTM nodes and is excluded from the candidate configuration. Although the one-layer LSTM with L1-200 and with L1-300 look like under-fitting (Fig. 16 Fig. 16(b) to Fig. 16(c)), they have an increasing trend in the training accuracy. Thus, they need more epoch to justify the training accuracy trend. Fig. 19 shows training accuracy and validation accuracy of the configurations for an extended epoch size 5000. Since the trends are saturated after epoch 3000 and do not look like under-fitting, they are included in the candidate configuration. Table 4 shows the training and validation accuracy trends for the candidate configurations along with standard deviation of accuracies. In order to check stability, standard deviations (σ) of accuracies in the last 100 epochs are added. The lower the standard deviation, the more stable the training accuracy and validation accuracy trends. Since they are quite stable, they can be candidate configurations.



(a) L1-200



(b) L1-300

Fig. 19. Accuracy and loss trends of SOD with one LSTM layer only having epoch size 5000

Table 4. Training and validation accuracies for candidate configurations

LSTM configuration	Training Accuracy %	Validation Accuracy %	σ for last 100 epochs	
			training accuracy	validation accuracy
L1-100	85.47	96.73	0.00351	0.00199
L1-200	95.28	96.05	0.00409	0.00043
L1-300	95.02	96.35	0.00228	0.00038
L1-100, L2-50	**97.40**	**96.80**	0.01407	0.00466

All the three stacked LSTM layers have unstable accuracy trends as shown in Fig. 18. Although configuration (L1-100, L2-75, L3-50) shows a stability near epoch 3000, some sparks can be expected from the accuracy trends. Other three stacked configurations tell additional layers and nodes do not help for the SOD performance. Thus, we did not experiment for the larger number of layers and nodes further.

Table 4 shows that two-layered configurations (L1-100, L2-50) has the highest accuracy in both training and validation. Thus, we decided that the configuration with two LSTM layers (L1-100 and L2-50), batch size 2048, epoch size 2000 and time step 4 gives us the best training and validation accuracy results in the experimental environment.

The followings are the minor hyper-parameters selected for configuration. The dropout rate 0.2 is selected only once after the second LSTM layer. Adding more dropout layers disturbs accuracy and loss trends. Similarly, increasing or decreasing dropout rate also disturb accuracy and loss trends. The Adam optimizer is used during training with a learning rate of 0.0001, because the smaller learning rate shows better results in the case.

4.3 Comparison with Other Methods

We compared SOD with state-of-the-art machine learning techniques. Logistic Regression 2-class classifier (LR) [14], Decision Trees classifier (C4.5) [32], Random Forest classifier (RF) [16], K-Neighbors classifier (KNN) [17], Multilayer Perceptron (MLP) [18], Support Vector classifier (SVC) [19], XG Boost classifier (XGB) [20], Cat Boost classifier (CB) [21], and Light GBM classifier (LGBM) [22] were trained on the same dataset for the comparison.

Scikit Learn library [33] was used to import decision tree classifier (C4.5) and Random Forest classifier in the python script. Both classifiers were trained using training and validation dataset. Decision tree shows training accuracy of 99.9% and validation accuracy of 84.72%, while Random Forest classifier shows training accuracy and validation accuracy of 99.8% and 79.06%, respectively. Logistic Regression classifier showed a training accuracy of 99.01% and validation accuracy of 92.58%, while K-Neighbors classifier with 3 neighbors showed training accuracy and validation accuracy of 99.91% and 85.27% respectively. SVC showed training accuracy of 99.68% and validation accuracy of 79.03%.

The gradient boosting based LightGBM, XG boost and Cat boost framework uses tree dependent learning algorithms. LightGBM classifier, XG boost classifier, and Cat boost classifier were imported in python script using LightGBM library, XG boost library and Cat boost library, respectively. The hyper-parameter for LightGBM and XG boost i.e. 31 number of leaves, 0.5 learning rate and traditional gradient boosting decision tree were used for training. In the case of Cat boost, its default gradient boosting decision trees are used. LightGBM classifier showed a training accuracy of 82.57% and validation accuracy of 82.34%, while XG boost classifier showed training accuracy and validation accuracy of 99.9% and 78.61% respectively. Cat boost showed training accuracy of 99.9% and validation accuracy of 81.74%.

Multilayer Perceptron (MLP) was used in python script using Scikit learn neural network library. Two hidden layers were used in MLP, the first and the second hidden layer consists of 100 and 50 neurons, respectively. Hyperbolic Tangent (Tanh) activation function and Adam optimizer with a learning rate of 0.0001 and 2000 number of iterations were used during MLP training. After training, MLP classifier showed a training accuracy of 96.35% and validation accuracy of 92.34%.

After successful training of SOD and other comparative models, the test dataset was fed to them. The results of test accuracy are shown in Table 5. H&C means the LSTM occupancy detection system proposed by Husnain and Choe [26]. The proposed LSTM model shows better performance and achieves the testing accuracy of 95.62%.

Table 5. Comparison between model performances

Model	Testing Accuracy (%)
C4.5	75.34
RF	71.78
KNN	79.39
SVC	68.04
LR	78.58
CB	74.83
LGBM	75.39
XGB	71.75
MLP	84.23
H&C	85.00
SOD	95.62

5 Conclusions and Future Works

This paper states the occupancy detection scheme to determine if there is any individual in a specific space. In homes and offices, rooms are often lighted up assuming max occupancy and lead the loss of electrical energy. We developed an occupancy detection system comprising a 4×4 thermal sensor array and a PIR sensor with LSTM module. Unlike the previous researches that detect occupancy in front of the thermal sensing area only, our proposed system SOD focuses on occupancy detection of an overall room including sensor off-range areas. The proposed deep learning model consists of a LSTM module and is able to memorize sequence pattern of human movements.

The proposed system shows better performance compared to state-of-the-art machine learning techniques. SOD system approach achieves testing accuracy of 95.62%, while the other techniques including Logistic Regression (2-class classifier) show relatively low test accuracies. The ideal occupancy detection system must be low-cost, reliable, privacy conserving and power efficient. We present an occupancy detection system that satisfies these requirements: two low cost and low power sensors, a robust LSTM network and no camera module.

SOD system is designed to resist multiple environmental noises like moving pets, warmed monitor, and warmed area by sun light. Since heat pattern of pets are quite different with one of human pattern, it is assumed that distinguish between pet and human is simple. Also, working monitors, a heater, or warmed areas have distinguishable patterns such that the areas are little changed during a short time. We have a plan to extend experimental environments for the noises and will test the system on various situations. Another possible expansion of SOD system is an intrusion detection for home / office security. By registering and learning movement patterns of residents, intruder can be distinguished by the pattern which does not match with any registered patterns.

Acknowledgements

This research was supported by Kumoh National Institute of Technology (202001250001).

References

[1] EIA, How much electricity is used for lighting in the United States?, U.S. Energy Information Administration (EIA). <https://www.eia.gov/tools/faqs/faq.php?id=99&t=3>, 2018 (accessed 19.11.21).

[2] L.R. Medsker, L.C. Jain, Recurrent Neural Networks Design and Applications, CRC Press, 2001.

[3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9(8)(1997) 1735-1780.

[4] Omron, Omron Electronic Components Europe. <http://components.omron.eu/Product-details/D6T-44L-06>, 2019 (accessed 19.11.28).

[5] S. Narayana, R.V.V. Prasad, V.S.S. Rao, T.V.V. Prabhakar, S.S.S. Kowshik, M.S.S. Iyer, PIR sensors: Characterization and novel localization technique, in Proc. the 14th International Conference on Information Processing in Sensor Networks, 2015.

[6] Z. Chen, C. Jiang, L. Xie, Building occupancy estimation and detection: A review, Energy Build. 169(2018) 260-270.

[7] Y. Agarwal, B. Balaji, S. Dutta, R.K.K. Gupta, T. Weng, Duty-cycling buildings aggressively: The next frontier in HVAC control, in: Proc. IPSN'11, 2011.

[8] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Anthony, E. Field, K. Whitehouse, The smart thermostat: using occupancy sensors to save energy in homes, in Proc. the 8th ACM Conference on Embedded Networked Sensor Systems, 2010.

[9] J. Scott, A.J. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, N. Villar, PreHeat: controlling home heating using occupancy prediction, in Proc. 2011 ACM Conference on Ubiquitous Computing, 2011.

[10] B. Qolomany, A. Al-Fuqaha, D. Benhaddou, A. Gupta, Role of deep LSTM neural networks and Wi-Fi networks in support of occupancy prediction in smart buildings, in Proc. 2017 IEEE 19th International Conference High Performance Computer Communication, 2018.

[11] K. P. Lam, M. Höynck, B. Dong, B. Andrews, Y.-Shang. Chiou, R. Zhang, D. Benitez, J. Choi, Occupancy detection through an extensive environmental sensor network in an open-plan office building, in Proc. 2009 International Building Performance Simulation Association, 2009.

[12] S. Mamidi, Y.H. Chang, R. Maheswaran, Improving building energy efficiency with a network of sensing, learning and prediction agents, in: Proc. 11th International Conference on Autonomous Agents and Multiagent Systems 2012: Innovative Applications Track, 2012.

[13] I. Amin, A.J. Tahylor, F. Junejo, A. AI-Habaibeh, R. Parkin, Automated people-counting by using low-resolution infrared and visual cameras, Meas. J. Int. Meas. Confed. 41(6)(2008) 589-599.

[14] H.F. Yu, F.L. Huang, C.J. Lin, Dual coordinate descent methods for logistic regression and maximum entropy models, Mach. Learn. 85(1-2)(2011) 41-75.

[15] J.R. Quinlan, C4. 5 Programs for Machine Learning, Elsevier, 2014.

[16] L. Breiman, Documentation for R package randomForest, Mach. Learn. 45(1)(2001) 5-32.

[17] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. 46(3)(1992) 175-185.

[18] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Journal of Machine Learning Research (9)(2010) 249-256.

[19] H.T. Lin, C.J. Lin, R.C. Weng, A note on Platt's probabilistic outputs for support vector machines, Mach. Learn. 68(3)(2007) 267-276.

[20] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in Proc. International Conference on Knowledge Discovery and Data Mining, 2016.

[21] A.V. Dorogush, V. Ershov, A. Gulin, CatBoost: gradient boosting with categorical features support. <https://arxiv.org/abs/1810.11363>, 2018.

[22] C.T. Dang, M. Straub, E. André, Hand distinction for multi-touch tabletop interaction, in: Proc. Advances in Neural Information Processing Systems, 2009.

[23] 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.

[24] A. Beltran, V.L. Erickson, A.E. Cerpa, ThermoSense: Occupancy thermal based sensing for HVAC control, in Proc. the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, 2013.

[25] A. Tyndall, R. Cardell-Oliver, A. Keating, Occupancy estimation using a low-pixel count thermal imager, IEEE Sens. J. 16(10)(2016) 3784-3791.

[26] H. Asif, T-Y. Choe, Smart occupancy detection system for efficient lighting using deep learning, in Proc. World IT Congr., 2019.

[27] Arduino, Arduino Products. <https://www.arduino.cc/en/Main/Products>, 2019 (accessed 19.09.23).

[28] Raspberry Pi, Raspberry Pi 3 Model B. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>, 2019 (accessed 19.09.23).

[29] PySerial, pySerial - pySerial 3.4 documentation. <https://pyserial.readthedocs.io/en/latest/pyserial.html>, 2019 (accessed 19.09.23).

[30] Python Foundation, About PythonTM | Python.org. <https://www.python.org/about/>, 2016 (accessed 19.09.23).

[31] Apache, Apache Cassandra0 <https://academy.datastax.com/planet-cassandra/what-is-apache-cassandra>, 2014 (accessed 19.09.23).

[32] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, Journal of machine learning research 12(2011) 2825-2830.