INTERNATIONAL JOURNAL OF APPLIED METHODS IN ELECTRONICS AND COMPUTERS

www.ijamec.org

International Open Access

Volume 13
Issue 02
June, 2025

Research Article

Room Occupancy Prediction from Temperature Data with Deep Convolutional Neural Networks

Özcan ÇATALTAŞ [a,*]

[a] *Electrical Electronic Engineering Department, Selcuk University, Konya, Türkiye*

ARTICLE INFO

Article history:
Received 14 April 2025
Accepted 16 June 2025

Keywords:
Convolutional neural networks
Machine learning
Occupancy detection

ABSTRACT

This study aims to estimate the number of people in a room using data from temperature sensors placed in a room. The study utilizes an open-source dataset comprising time-dependent temperature sensor data. The days when the number of people in the dataset was always zero were removed to avoid misleadingly increasing the success of the model, and the number of data points was reduced by averaging every 10 measurements. The temperature data were converted into RGB images of 28 x 28 pixels, and the measurements from each sensor were assigned to a different region in the image. A convolutional neural network model was trained by dividing these images into training, validation, and test sets. The model was able to predict the no-person and low-person classes with high accuracy. However, at higher headcounts, the model's performance degraded. In particular, prediction errors increased in transition situations where the number of people changes rapidly. The accuracy of the model on the test dataset is obtained as 93.33%. The results show that temperature data can be effectively used to predict occupancy levels. This study lays a foundation for headcount prediction based on temperature data and offers significant potential in applications such as smart building systems.

1. Introduction

Occupancy detection is the process of determining the presence of people within a building or a specific area. This process can range from simply understanding whether a space is occupied or unoccupied to estimating the number of people present [1], [2]. The importance of occupancy detection stems from the fact that it plays an important role in the energy performance of buildings and users' perception of the indoor environment. People are one of the main drivers of energy use in buildings and also one of the leading causes of the gap between the expected and actual performance of buildings. Therefore, it is vital to optimize building operations based on actual or estimated occupancy [3].

Literature has shown that an average of 23% energy savings can be achieved when using occupancy-based demand-side heating, ventilation, and air conditioning (HVAC) systems. Energy-saving potential can vary depending on site characteristics, utility, and occupant behavior. Energy-saving potentials can be estimated by matching occupancy information with building energy simulations [3].

The benefits of occupancy detection are not limited to energy savings [4]. It also plays a critical role in optimizing resource allocation, security measures, and user-centric services. Accurate occupancy prediction improves energy efficiency by ensuring that building systems such as lighting, heating, and ventilation are operated only when needed[5]. This reduces operating costs and environmental impact. Occupancy information can also be valuable for building security systems, for example, to detect unauthorized entry or to determine the number of people inside a building during emergencies. In terms of occupant comfort, automatically adjusting systems to people's presence and preferences provides a better indoor experience[6].

Various sensors and hardware are used for occupancy

* Corresponding author. E-mail address: *ozcancataltas@selcuk.edu.tr*
DOI: 10.58190/ijamec.2025.121

detection [3]. A review of the literature reveals that various combinations of features have been explored. The most commonly used features are temperature, followed by CO2 concentration, relative humidity, light and sound pressure. In addition, motion sensors [7], infrared array sensors [8], cameras, Wi-Fi routers [9], Bluetooth low-energy devices, ultra-wideband radar [10], and even electricity meters [11] are among the technologies used for occupancy detection.

Artificial intelligence (AI) and machine learning (ML) algorithms play an important role in occupancy detection [2]. Thanks to their ability to model complex relationships of data from various sensors and make accurate predictions, ML techniques have significantly improved the accuracy and efficiency of occupancy detection systems. In the literature, numerous machine learning (ML) algorithms have been evaluated for occupancy detection. Support Vector Machines, Artificial Neural Networks, Hidden Markov Models, Decision Trees, Random Forests, Gradient Boosting Machines, K-nearest Neighbor, Convolutional Neural Networks, and Long Short-Term Memory Networks are some of them. Studies show that high accuracy rates can be achieved primarily by using various sensor combinations and ML algorithms [1], [2], [4], [12], [13].

In this study, a deep neural network model was created using data from four temperature sensors placed in specific regions of a room to classify the number of occupants. Unlike many previous studies that rely on multiple sensor types or complex sensor networks, the main motivation of this research is to investigate whether accurate occupancy detection and classification can be achieved using only temperature data, which is both cost-effective and easy to implement. This approach offers a novel contribution to the literature by demonstrating that a simplified sensor setup can still yield high prediction accuracy, potentially lowering the barriers to practical adoption in real-world building environments. By focusing solely on temperature sensors, this study provides new insights into the feasibility of minimalistic and scalable occupancy detection solutions, which can facilitate broader applications in energy management and smart building systems.

2. Material and Methods

2.1. Dataset Description

This study uses the dataset created by A. Singh et al. [2]. This dataset is collected through a wireless sensor network to estimate the number of people in a room. This network consists of seven low-cost sensor nodes configured in a star topology, and the sensors are deployed in an office environment of approximately 24 square meters. During data collection, measurements were taken from CO2, temperature, light, sound, and motion sensors. The dataset contains a total of 10129 samples collected on seven different days. The number of occupants in the room is given between 0 and 3. The dataset used is available as open source from the UC Irvine Machine Learning Repository [14].

The temperature sensor data at four different points in this dataset, which are the subject of our study, were collected, and data from other sensors were not included in the study. The location of the temperature sensors in the room is shown in Figure 1.

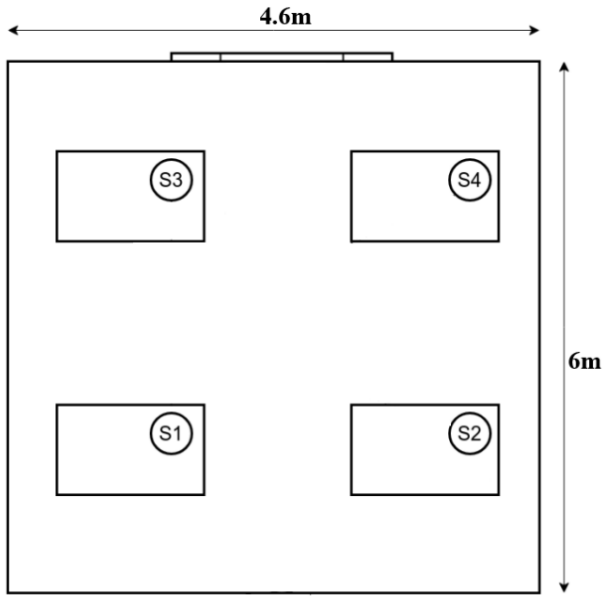


Figure 1. Sensor placement in the room (reproduced from [2]).

2.2. Data Preprocessing

The dataset used in this study consists of time-dependent measurements from temperature sensors in a room. The dataset covers a total of seven days of measurements and contains a data record every 30 seconds. However, for four days in the dataset, the number of people was always recorded as zero because there was no one in the room. Since this could misleadingly overestimate the success of the model, the data for these days were excluded from the analysis. As a result of this process, the total number of samples was reduced, yielding a more balanced dataset.

The data recorded for every 30 seconds in the dataset was resampled to reduce the number of data points and optimize the processing time. During this process, the dataset was reduced by a factor of 10 by averaging every 10 data points. However, it should be noted that the time intervals were not precisely equal during this process. As a result of the resampling process, the total number of samples was reduced to 525. Figure 2 shows the data for one day and one temperature sensor before and after the reduction process.

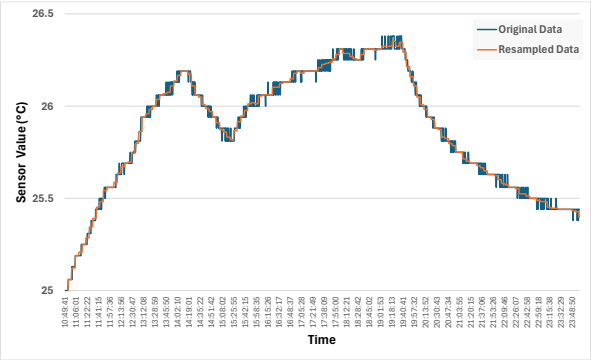


Figure 2. Original and resampled sensor data

The temperature data was converted into RGB images to be used as input data for the model. During this process, the temperature values were normalized to a range of 24-30°C and scaled to 0-1. The normalized temperature values were used to create images with a size of 28x28 pixels. The measurements from four different temperature sensors in the dataset were assigned to four different regions on the image. Each sensor represents an area of 14 x 14 pixels in the 28 x 28 pixel image. The "jet" color map was used for color coding in the creation of the images. This color map made the temperature values visually more easily distinguishable. Sample images obtained as a result of the transformation are given in Figure 3.

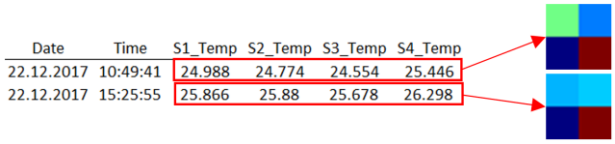


Figure 3. Image generating process.

Finally, the generated images were divided into training, validation, and test sets. The dataset was randomly divided into 60% training, 20% validation, and 20% testing. This was done to evaluate the performance of the model and to test its generalization ability. The statistical information of the groups obtained after the division process is given in Table 1.

Table 1. The statistical information about the Train, Validation and Test sets.

	Count	Sensor-1 Avg±Std	Sensor-2 Avg±Std	Sensor-3 Avg±Std	Sensor-4 Avg±Std
Train	315	25.63±0.40	25.80±0.75	25.24±0.48	25.87±0.36
Validation	105	25.65±0.36	25.79±0.58	25.26±0.41	25.90±0.33
Test	105	25.64±0.38	25.82±0.70	25.27±0.47	25.86±0.34

2.3. Model Architecture

In this study, a Deep Convolutional Neural Network model is used to predict the number of people in a room. The CNN model is widely used in classification problems due to its ability to extract features and learn complex patterns from image data automatically [15], [16], [17]. The architecture of the model consists of an input layer, convolutional layers, pooling layers, fully connected layers, and a classification layer. The block diagram of the model used is given in Figure 4.

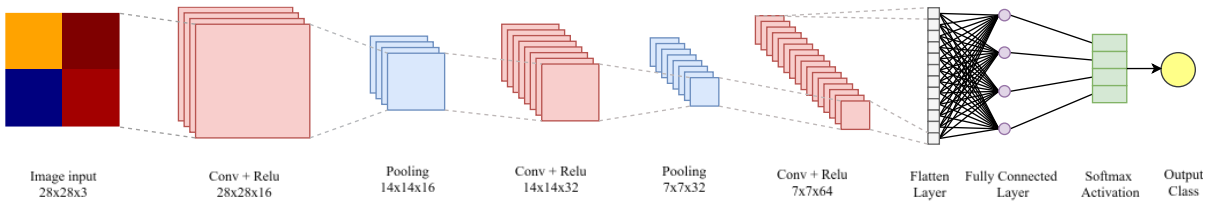


Figure 4. Model Architecture

The input layer of the model is designed to receive images with a size of 28 x 28 pixels and three color channels (RGB). This layer passes the image data to the other layers of the model.

There are three convolution layers in the model. Each convolution layer uses filters of size 3x3 and maintains the input size with "same" padding. The first convolution layer has 16 filters, the second convolution layer has 32 filters, and the third convolution layer has 64 filters. The convolution layers are used to learn local features in the image. A Batch Normalization layer and the ReLU activation function follow each convolution layer. Batch Normalization helps to train the model in a faster and more stable way, while the ReLU activation function adds a non-linear structure, allowing the model to learn more complex patterns. After the convolution layers, we used max pooling layers. Max pooling layers help the model learn more general features and reduce computational cost by reducing the size of the feature maps. After the first two convolution layers, max-pooling layers of size 2x2 and stride two are used.

At the end of the model, there is a fully connected layer. This layer performs the classification process by combining the features obtained from the convolution and pooling layers. The fully connected layer has as many neurons as the number of classes. In this study, the number of classes represents the different categories of person counts in the dataset. After the fully connected layer, the softmax activation function is used to obtain the probability values for each class. The number of trainable

parameters in the model is 36356.

Finally, the classification layer, which is the output layer of the model, determines to which class the input image belongs by using the probability values obtained from the softmax layer.

2.4. Training Process

The training of the model was performed on a dataset created to predict the number of people in a room. During the training process, various methods and hyperparameters were employed to optimize the model's parameters and enhance its generalization ability. To evaluate the model's performance and test its generalization ability, the dataset was divided into three groups: a training set (60%), a validation set (20%), and a test set (20%). The training set is used during the model's learning process, while the validation set is used to monitor the model's performance during training and to mitigate the risk of overfitting. The test set is reserved for evaluating the overall performance of the model after training is completed.

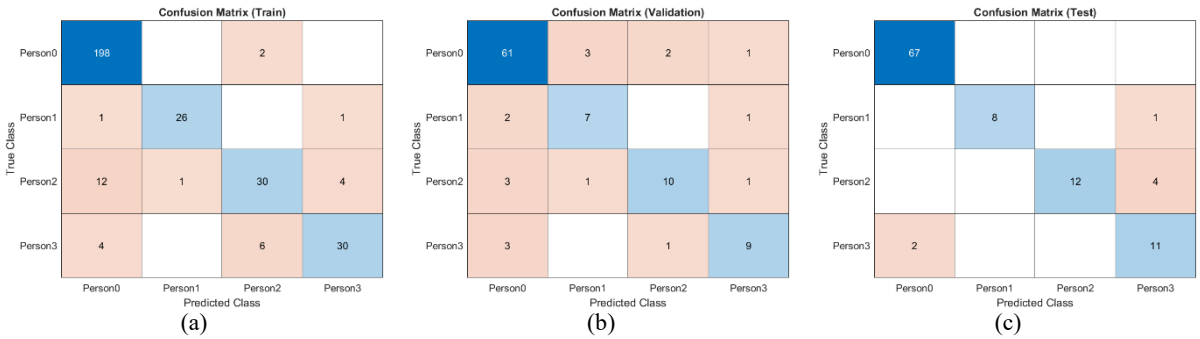
The Adam optimization algorithm was used to train the model. Adam provides a faster and more stable learning process by dynamically adjusting the learning rate. The initial learning rate was set to 0.001, which yielded a stable speed during model parameter optimization. The maximum number of epochs was set to 100, but thanks to the early stopping mechanism, the training process was terminated early when the validation loss did not improve. The mini-batch size was set to 128, and 128 images were fed to the model during each iteration. The training process was carried out by optimizing the model on the data in the training set. The model's performance was evaluated on the validation set at the end of each epoch. The validation loss was used as a metric to monitor the generalization ability of the model. If the validation loss did not improve for a certain period, the early stopping mechanism was activated, and the training process was terminated. This method prevented the model from overfitting and increased its generalization ability.

2.5. Performance Metrics

The performance of the model is evaluated on the test dataset using various metrics to analyze its success in detail in predicting the number of people in a room. Among them, the confusion matrix is used to visualize the classification performance of the model and analyze the correct and incorrect predictions for each class; it presents in tabular form the relationship between the actual classes and the classes predicted by the model and allows examination in detail in which classes the model is more successful and in which classes it makes mistakes. Accuracy refers to the proportion of instances correctly predicted by the model to the total number of instances in the test dataset and is used as a key measure to evaluate the overall performance of the model. The F1 score is the harmonic mean of the model's precision and recall and is important for evaluating the model's performance on unbalanced datasets. It measures the model's ability to make correct positive predictions and its success in minimizing false negative predictions. Sensitivity refers to the rate at which the model correctly predicts true positive instances and measures how well the model can detect instances belonging to a class. Specificity refers to the rate at which the model correctly predicts true negative instances and measures the model's success in correctly excluding instances that do not belong to a class.

3. Results and Discussion

In this study, the performance of a deep learning model developed to predict the number of people in a room is evaluated using metrics such as sensitivity, specificity, and F1 score on training, validation, and test datasets. Figure 5 shows the confusion matrices obtained as a result of classification, and Table 2 shows the performance metrics for each dataset.



(a) (b) (c)

Figure 5. Confusion Matrices for Training, Validation, and Test Datasets

Table 2. Model Performance Metrics: Sensitivity, Specificity, and F1 Score Values on Training, Validation, and Test Datasets. All values are given in percentages.

Set	Metric	CLASS			
		Occupancy Level-0	Occupancy Level-1	Occupancy Level-2	Occupancy Level-3
Train	Sensitivity	99.00	92.86	63.83	75.00
	Specificity	85.22	98.95	97.76	98.18
	F1 Score	95.42	91.23	72.29	80.00
	Accuracy	90.16			
Validation	Sensitivity	91.04	70.00	66.67	64.29
	Specificity	79.49	94.79	96.70	96.74
	F1 Score	89.71	63.64	71.43	69.23
	Accuracy	82.86			
Test	Sensitivity	100	88.89	75.00	84.62
	Specificity	94.74	100	100	94.57
	F1 Score	98.53	94.12	85.71	75.86
	Accuracy	93.33			

The training set results show that the model performs well in the learning process. In particular, the sensitivity is 99%, the specificity is 85.22%, and the F1 score is 95.42% for the Occupancy Level-0 class, which represents situations where no person is present. This indicates that the model's ability to detect unoccupied situations accurately is quite strong. Similarly, the sensitivity was 92.86%, the specificity was 98.95%, and the F1 score was 91.23% for the Occupancy Level-1 class, which represents a low number of people. However, in Occupancy Level-2 and Occupancy Level-3 classes with higher person counts, the sensitivity remained at lower levels of 63.83% and 75.00%, respectively. This indicates that the model struggles in the learning process at higher person counts and needs further improvement for these classes.

The validation set results were used to evaluate the generalization ability of the model. For the Occupancy Level-0 class, the sensitivity was 91.04%, the specificity was 79.49%, and the F1 score was 89.71%. This indicates that the model can accurately detect non-person cases in the validation set. However, in the Occupancy Level-1 class, which represents a low number of people, the sensitivity was 70.00%, and the F1 score was 63.64%. In Occupancy Level-2 and Occupancy Level-3 classes, which have higher person counts, the sensitivity is calculated as 66.67% and 64.29%, respectively. These results show that the model struggles in the validation set at higher person counts, and its generalization ability is limited. However, the high specificity values for all classes indicate that the model has a low tendency to make false positive predictions and can discriminate well between classes.

The test set results were used to evaluate the performance of the model on real-world data. For the Occupancy Level-0 class, the sensitivity was 100%, the specificity was 94.74%, and the F1 score was 98.53%. This shows that the model perfectly detects situations where no person is present in the test set. For the Occupancy Level-1 class, which represents a low number of people, the sensitivity is 88.89%, the specificity is 100%, and the F1 score is 94.12%. However, for Occupancy Level-2 and Occupancy Level-3 classes with higher person counts, the sensitivity was 75.00% and 84.62%, respectively. This shows that the performance of the model decreases in the test set at higher person counts. F1 scores are calculated as 85.71% and 75.86% for these classes, respectively.

In general, the model is quite successful in detecting no-person cases and low-person count classes, but the performance decreases at higher person counts. This may be due to the class imbalance in the dataset. In particular, the availability of less data for higher person counts may have limited the model's ability to learn these classes. The decrease in sensitivity and F1 scores indicates that the model has an increased tendency to make false negative predictions at higher person counts. However, the specificity values are generally high, suggesting that the model has a low tendency to make false positive predictions and can discriminate well between classes.

Figure 6 shows the predicted and actual people count for each day. When the forecasting performance of the model is analyzed on a day-by-day basis, it is observed that the model can accurately capture general trends, but the forecast errors increase in cases of sudden changes. While the model's predictions are quite close to the actual values in periods when the number of people is constant, deviations in the model's predictions occur in transition situations where the number of people increases or decreases rapidly. In particular, the performance of the model was quite high when there were no people, and generally, successful predictions were made at low headcounts. However, at higher numbers of people, a significant drop in the model's prediction accuracy was observed. This suggests that the model's ability to learn transition situations is limited.

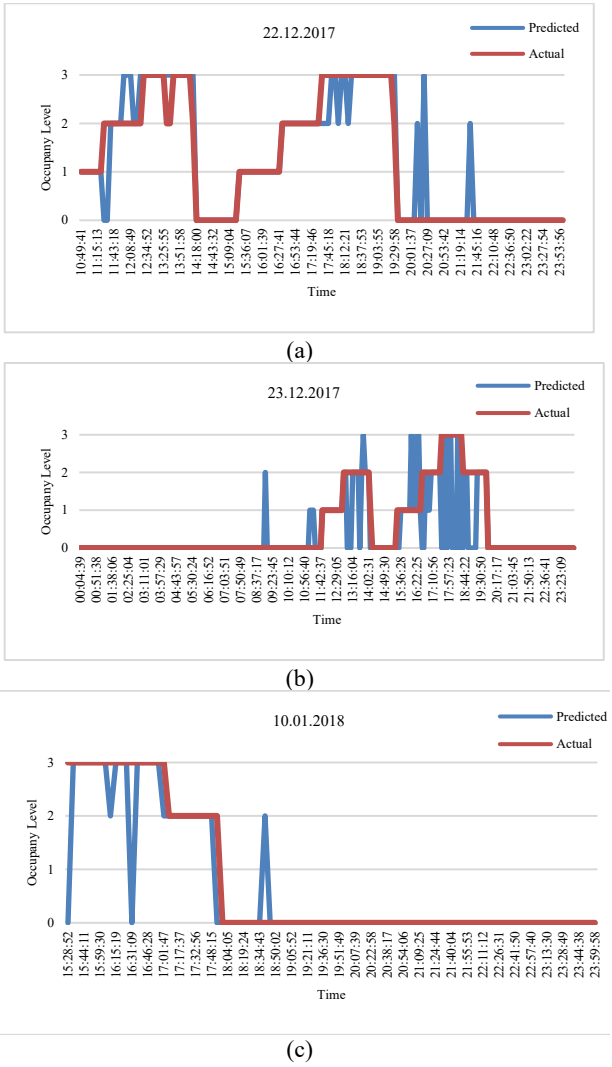


Figure 6. Graphical representation of predicted and actual number of people for each day

As a result, the model seems to be successful in the task of predicting the number of people, but additional improvements are needed to improve performance in situations with sudden changes. The results show that temperature data can be used to predict the number of people. However, the dataset needs to be expanded, and the class imbalance needs to be addressed to improve the performance of the model at high headcounts. In the future, the generalization capability of the model can be more thoroughly evaluated by testing in different rooms and with a larger number of sensors.

4. Conclusions

In this study, a deep learning model was developed to predict the number of people in a room using data from temperature sensors in a room. The model performed well on the task of predicting the number of people using only temperature data, achieving an accuracy of 93.33% on the test set. Evaluations on training, validation, and test datasets showed that the model was able to detect no-person situations and low-person count classes accurately. However, at higher person counts, a significant drop in the model's performance was observed. This may be due to the class imbalance in the dataset and the limited availability of data for high-headcount classes.

Day-by-day analyses reveal that the model accurately captures general trends, but the prediction errors increase in transitional situations when the headcount changes rapidly. In particular, while the model's predictions were quite close to the actual values in periods when the number of people was constant, there were deviations in the model's performance when there were sudden changes.

The results of this study show that temperature data can be used to predict headcount and that such an approach has significant potential for energy savings and resource management in applications such as smart building systems. However, in order to improve the performance of the model, the dataset should be expanded, and the class imbalance problem should be addressed. Furthermore, testing the model on different rooms, environments, and sensor placements will allow a more comprehensive evaluation of its generalization capability.

Declaration of Ethical Standards

The article does not contain any studies with human or animal subjects.

Credit Authorship Contribution Statement

The author individually was responsible for the ideation, modeling, analysis, and writing of this article.

Declaration of Competing Interest

The author claims that there are no conflicts of interest.

Funding / Acknowledgements

No funding was received from any organization for this research.

References

[1] X. Dai, J. Liu, and X. Zhang, "A review of studies applying machine learning models to predict occupancy and window-opening behaviours in smart buildings," Energy Build, vol. 223, p. 110159, Sep. 2020, doi: 10.1016/J.ENBUILD.2020.110159.

[2] A. P. Singh, V. Jain, S. Chaudhari, F. A. Kraemer, S. Werner, and V. Garg, "Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes," 2018 IEEE Globecom Workshops, GC Wkshps 2018 - Proceedings, Jul. 2018, doi: 10.1109/GLOCOMW.2018.8644432.

[3] F. Banihashemi, M. Weber, F. Deghim, C. Zong, and W. Lang, "Occupancy modeling on non-intrusive indoor environmental data through machine learning," Build Environ, vol. 254, Apr. 2024, doi: 10.1016/j.buildenv.2024.111382.

[4] R. Mahto and K. Sood, "Advancing Occupancy Detection through Deep Learning and Sensor Integration," 2024 IEEE 14th Annual Computing and Communication Workshop and Conference, CCWC 2024, pp. 173–177, 2024, doi: 10.1109/CCWC60891.2024.10427586.

[5] K. Tutuncu, M. Koklu, and O. Cataltas, "Occupancy Detection Through Light, Temperature, Humidity and CO2 Sensors Using ANN." [Online]. Available: http://iraj.in

[6] M. Emad-Ud-Din and Y. Wang, "Indoor Occupancy Sensing via Networked Nodes (2012–2022): A Review," Mar. 01, 2023, MDPI. doi: 10.3390/fi15030116.

[7] I. P. Mohottige and T. Moors, "Estimating Room Occupancy in a Smart Campus using WiFi Soft Sensors," Proceedings -

Conference on Local Computer Networks, LCN, vol. 2018-October, pp. 191–199, Jul. 2018, doi: 10.1109/LCN.2018.8638098.

[8] K. Hashimoto, K. Morinaka, and N. Yoshiike, "People count system using multi-sensing application," International Conference on Solid-State Sensors and Actuators, Proceedings, vol. 2, pp. 1291–1294, 1997, doi: 10.1109/SENSOR.1997.635472.

[9] L. Rueda, K. Agbossou, A. Cardenas, N. Henao, and S. Kelouwani, "A comprehensive review of approaches to building occupancy detection," Build Environ, vol. 180, p. 106966, Aug. 2020, doi: 10.1016/J.BUILDENV.2020.106966.

[10] Z. Chen, C. Jiang, and L. Xie, "Building occupancy estimation and detection: A review," Energy Build, vol. 169, pp. 260–270, Jun. 2018, doi: 10.1016/J.ENBUILD.2018.03.084.

[11] A. U. Nambi, A. Irawan, A. Nurhidayat, B. Humala, and T. R. Dharmawan, "Predicting room-level occupancy using smart-meter data," International Journal of Distributed Systems and Technologies, vol. 8, no. 4, pp. 1–16, Oct. 2017, doi: 10.4018/IJDST.2017100101.

[12] B. Dong et al., "An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network," Energy Build, vol. 42, no. 7, pp. 1038–1046, Jul. 2010, doi: 10.1016/J.ENBUILD.2010.01.016.

[13] Z. Yang, B. Becerik-Gerber, N. Li, and M. Orosz, "A systematic approach to occupancy modeling in ambient sensor-rich buildings," Simulation, vol. 90, no. 8, pp. 960–977, 2014, doi: 10.1177/0037549713489918.

[14] "Room Occupancy Estimation - UCI Machine Learning Repository." Accessed: Apr. 04, 2025. [Online]. Available: https://archive.ics.uci.edu/dataset/864/room+occupancy+estimation

[15] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," Neural Comput, vol. 29, no. 9, pp. 2352–2449, Sep. 2017, doi: 10.1162/NECO_A_00990.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Accessed: Apr. 10, 2025. [Online]. Available: http://code.google.com/p/cuda-convnet/

[17] N. YÜCEL and M. YILDIRIM, "Classification of tea leaves diseases by developed CNN, feature fusion, and classifier based model," International Journal of Applied Methods in Electronics and Computers, vol. 11, no. 1, pp. 30–36, Mar. 2023, doi: 10.18100/IJAMEC.1235611.