



Ubiquitous multi-occupant detection in smart environments

Daniel Fährmann¹ · Fadi Boutros¹ · Philipp Kubon¹ · Florian Kirchbuchner¹ · Arjan Kuijper^{1,2} · Naser Damer^{1,2}

Received: 14 December 2022 / Accepted: 20 October 2023 / Published online: 27 November 2023
© The Author(s) 2023

Abstract

Recent advancements in ubiquitous computing have emphasized the need for privacy-preserving occupancy detection in smart environments to enhance security. This work presents a novel occupancy detection solution utilizing privacy-aware sensing technologies. The solution analyzes time-series data to detect not only occupancy as a binary problem, but also determines whether one or multiple individuals are present in an indoor environment. On three real-world datasets, our models outperformed various state-of-the-art algorithms, achieving F1-scores up to 94.91% in single-occupancy detection and a macro F1-score of 91.55% in multi-occupancy detection. This makes our approach a promising solution for improving security in smart environments.

Keywords Human activity recognition · Machine learning · Pattern recognition · Safety

1 Introduction

Occupancy detection (OD) refers to detecting the presence and absence of occupants [1]. OD is not only of scientific interest, but has practical applications, as emphasised by the following examples. Improper operation of heating, ventilation and air-conditioning (HVAC) systems can waste large amounts of energy inside indoor environments

[2, 3]. Manually controlled HVAC systems do not properly adapt to changes in the environment. This eventually leads to situations in which the system is running and consuming energy, although nobody is present to benefit from it. High costs for electricity and negative economical and ecological implications are the consequences [3]. OD technologies can enhance human comfort by enabling automated HVAC systems that regulate climate depending on the presence of individuals detected in an indoor environment. Unauthorised access to restricted areas (e.g., office buildings or industrial facilities) can be a threat for infrastructure or the safety of individuals [4]. The ability to detect whether one or multiple individuals are present in restricted areas contributes to more secure smart environments.

Previous work on OD either discussed approaches for simple OD (i.e., the detection of an individual) [5–12] or multi-occupancy detection (i.e., determining whether one or multiple individuals are present) [13, 14]. Privacy-invasive sensors (i.e., cameras and microphones) [15, 16], as well as privacy-aware ambient sensors have been used to monitor indoor environments [5–14]. Privacy-aware ambient sensors are important because they preserve the privacy of individuals. Detection systems that are based on privacy-aware sensing technologies usually find greater public acceptance. Privacy-aware ambient sensors measure characteristics like temperature, humidity or carbon dioxide (CO₂) concentration. The time series data recorded by

✉ Daniel Fährmann
daniel.faehermann@igd.fraunhofer.de

Fadi Boutros
fadi.boutros@igd.fraunhofer.de

Philipp Kubon
philipp.kubon@gmail.com

Florian Kirchbuchner
florian.kirchbuchner@igd.fraunhofer.de

Arjan Kuijper
arjan.kuijper@igd.fraunhofer.de

Naser Damer
naser.damer@igd.fraunhofer.de

¹ Smart Living and Biometric Technologies, Fraunhofer Institute for Computer Graphics Research IGD, Fraunhoferstraße 5, Darmstadt 64283, Hesse, Germany

² Department of Computer Science, Technical University of Darmstadt, Hochschulstraße 10, Darmstadt 64283, Hesse, Germany

ambient sensors enables ubiquitous OD in smart living environments.

This work presents a novel solution for OD. Our solution uses time series data recorded only by privacy-aware ambient sensors to detect individuals. Our solution is based on a Bidirectional gated recurrent unit (BiGRU) architecture that takes the temporal dependency of successive sensory signals into consideration. Besides general OD, our proposed solution can differentiate whether one or multiple individuals occupy an indoor environment.

We organized the structure of this work as follows. Section 2 presents previous work on OD. Section 4 presents the OD solution proposed in this work, including the underlying Artificial Neural Network (ANN) architecture, as well as the definitions of the OD tasks. Section 5 presents details on the datasets we used for our experiments. Section 6 presents the experiments, including the dataset preprocessing procedures, as well as the ANN hyper-parameter configurations we used. In Sect. 7, we discuss the results and compare the detection performance of our solution to state-of-the-art algorithms. Finally, Sect. 8 presents the conclusion.

2 Related work

This section presents previous work on privacy-aware OD since the solution presented in this work is also based on privacy-aware sensing technology. We disregard work on privacy-invasive OD (i.e., methodologies that are based on privacy-invasive sensing technology), as this is outside the scope of this work.

Pirttikangas et al. [17] examined the challenges and technologies in occupancy sensing, emphasizing the importance of enhancing location accuracy and considering privacy aspects. Their work also highlighted the potential of sensor fusion techniques and identified emerging research areas in predictive user behavior analysis. Following the work of Pirttikangas et al. [17], which underscored the importance of privacy aspects, a variety of approaches for privacy-aware OD were suggested in previous work [5–14].

Dong et al. [5] created a comparatively big test environment, including a larger workplace with many rooms and floors. The researchers installed various types of ambient sensors including CO₂, carbon monoxide (CO), total volatile organic compounds (TVOC), fine particulate matter (PM_{2.5}), acoustic, illumination, motion, temperature and humidity sensors in the experimental environment. Using the sensors, the researchers recorded a dataset over several weeks and determined the best combination of features based on the relative information gain (RIG). Regarding their investigations, CO₂ and acoustic

information is the most important features. The selected features were used to train various machine learning models, including traditional and deep learning models to determine the number of occupants.

In [13], Han et al. proposed an approach to detect multiple individuals based on CO₂ measurements. The authors assumed that the amount of CO₂ generated in an indoor environment scales linearly with the number of occupants. They suggested the use of a dynamic ANN that processes time-delayed information. Their time-delayed neural network (TDNN) enables the extraction of features from time windows instead of just considering a single data point without context. The authors collected CO₂ data sampled in one-minute intervals over five days. Their experimental results show that the use of time-delay information enhances detection performance. The size of the time window considered was optimal at approximately 7 min. The authors observed time-delayed responses from their ANN architecture, which they explained by gas dispersion in space. However, the authors did not explore alternative architectures (e.g., RNNs) to cope with the time dependency of sensory signals. Nor did they investigate the influence of additional environmental sensors (e.g., temperature and humidity sensors) on predictive performance.

Alhamoud et al. [6] investigated OD and indoor localization in a home environment based on bluetooth technology. The authors established a bluetooth personal area network (PAN) using three bluetooth USB sticks (i.e., beacons) in different rooms. They configured a mobile phone to run a bluetooth discovery search periodically to record the received signal strength indicator (RSSI) data (i.e., signal strength) from all three beacons. By measuring signal strength, their approach recognized whether a person is present in the room. The authors investigated supervised, as well as unsupervised models, including k-means clustering, support vector machine (SVM), and an ANN. The authors showed that person detection and localization with bluetooth is feasible in their setting. However, their approach may not be applicable to every common home environment because it requires isolated rooms to avoid the influence of other bluetooth signal sources.

In [14], the authors explored OD based on CO₂ concentration. The authors did not apply machine learning models, but built a system based on differential equations that is driven by proxy sensing. However, their approach requires explicit knowledge about the location of the sensors.

Candanedo and Feldheim [7] conducted experiments using multiple types of environmental sensors. They investigated different combinations of temperature, humidity, light (i.e., brightness), and CO₂ sensory data to detect occupancy in a binary classification scenario. The authors created a dataset and investigated the performance

of several classification models, including classification and regression trees (CART), random forests (RF), gradient boosting machine (GBM), and linear discriminant analysis (LDA). However, the authors did not evaluate the performance of more complex ANN models.

In [8], Szczurek et al. proposed an approach to determine the number of occupants. The authors measured CO₂ concentration, temperature, and relative humidity using a single combined sensor device in an intermittently used university classroom. The authors compared the performance of k-Nearest Neighbors (k-NN) to LDA. Their experimental results show that OD performance can be improved using a combination of sensors, especially when combining temperature and humidity data.

Wang et al. [9] suggested an approach for determining the number of occupants inside an office room. The authors combined data from several environmental sensors with Wi-Fi information to increase detection performance. The environmental sensors were used to continuously measure CO₂ concentration, temperature, and relative humidity over three days. Wi-Fi connection requests and responses were captured based on MAC addresses and RSSI. The authors evaluated three different machine learning models, including k-NN, SVM, and an ANN, and investigated different combinations of sensory data. Their ANN performed best when using only environmental sensor data. When using only Wi-Fi data, their ANN performed less well.

Hoori et al. [10] proposed the multicolumn radial basis function network (MCRN) mechanism that improves upon the traditional radial basis function network (RBFN) algorithm. Their MCRN mechanism divides the training set of a dataset into smaller subsets, using the k-dimensional tree (k-d tree) algorithm. The authors also reported on the OD performance of several baseline algorithms.

Liu et al. [11] proposed their multivariate convolutional neural network (MVCNN) approach. The authors used their model for the classification of multivariate time series and applied it to the OD task.

In [12], the authors applied the double deep Q-learning (DDQN) reinforcement learning algorithm to multivariate time series classification scenarios, including occupancy and fall detection. The authors extended the DDQN algorithm using a prioritized experience replay (PER) strategy to improve detection performance in rare event classification tasks.

Earlier work in the field of multi-occupancy detection focused heavily on the use of CO₂ concentration to detect multiple individuals [13, 14]. However, the use of CO₂ concentration has a significant disadvantage, as it is strongly influenced when windows are open or when stoves are used. The solution presented in this work differs because it utilizes data from a variety of different ambient

sensors (excluding CO₂ concentration) for multi-occupancy detection.

3 Problem statement

OD in indoor environments is a crucial component for enhancing security and enabling smart living applications. However, deployments of OD technologies often face challenges such as invasion of privacy, computational overhead, and limited occupancy state resolution. This section outlines the problems addressed in this work.

Privacy-friendly sensing Many OD approaches rely on cameras and other invasive sensing technologies such as microphones [15, 16]. Such approaches often face resistance from occupants due to privacy concerns. A primary problem addressed in this work is an OD solution that respects the privacy of the occupants.

Occupancy state resolution Traditional OD approaches mainly focus on binary classification, i.e., whether a space is occupied or not [5–12]. However, in certain scenarios, it is essential to determine whether one or multiple individuals are present in a given space. The second problem this work addresses is the distinction between one and multiple occupants. Moreover, the complexity of this issue extends to a dataset that we have preprocessed in a novel manner to make it suitable for multi-occupant detection.

To address these problems, this work proposes a novel solution for multi-occupant detection, aiming to preserve the privacy of the occupants. This also encompasses the adaptation of a dataset, which we have innovatively preprocessed to make it suitable for multi-occupant detection experiments.

4 Methodology

This section presents our novel occupancy detector and its components.

Our solution is based on the BiGRU architecture. The BiGRU architecture is special because it considers the time dependency of successive signals from the ambient sensors, making it particularly well-suited to OD, especially for multi-occupancy detection, where the sensory signals caused by multiple individuals may be temporally separated.

In the following, we describe the processing pipeline of our OD solution, as visualized in Fig. 1. The first step involves preprocessing an input dataset. Next, the preprocessed data is fed into our BiGRU model, which accepts consecutive samples as input. In the final step, the occupancy verdict is obtained. Our detector can handle two

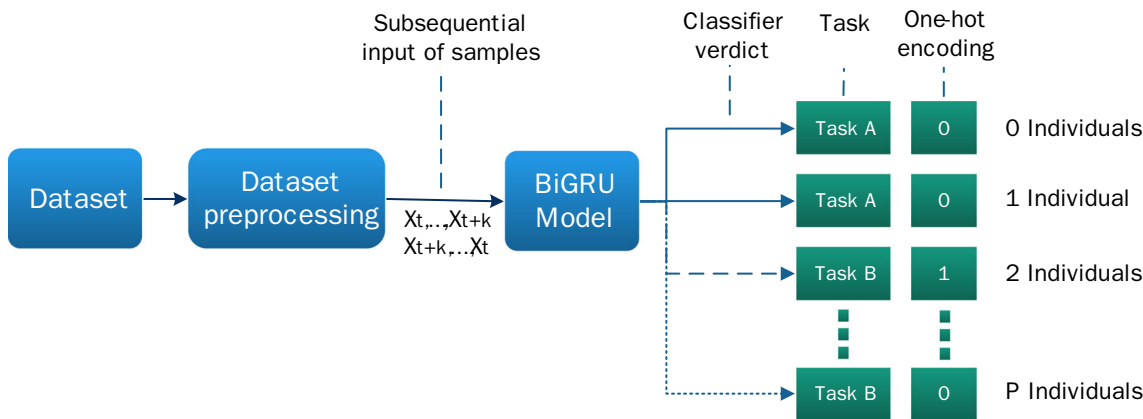


Fig. 1 The processing pipeline of our OD solution. After preprocessing a dataset, a sequence of consecutive samples is fed into our BiGRU model to obtain the occupancy verdict

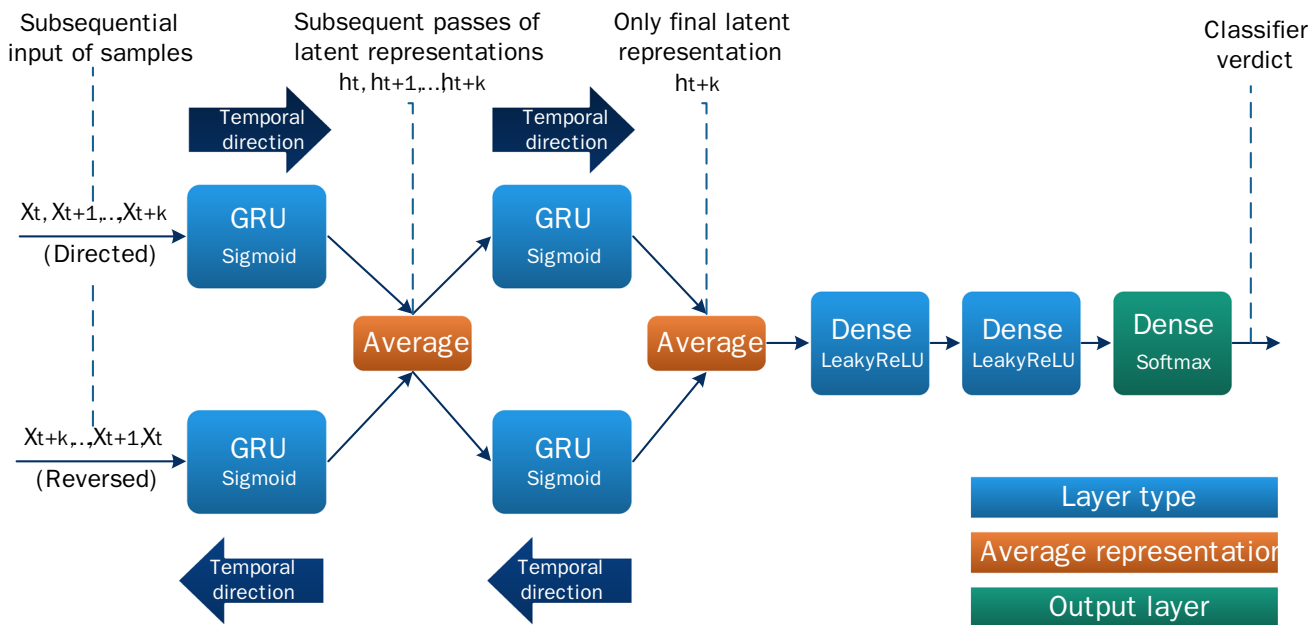


Fig. 2 The ANN architecture of our BiGRU occupancy detector consists of BiGRU layers, two fully connected layers, and a softmax output layer. The outputs of the BiGRU layers are combined by averaging

different tasks in indoor environments, defined as occupancy detection and multi-occupancy detection:

Definition 1 (Occupancy detection) refers to the detection of the presence of at least one person by means of ambient sensors in an indoor environment at a given time or time span.

Definition 2 (Multi-occupancy detection) refers to determining whether only one or multiple individuals are present in an indoor environment at a given time or time span.

4.1 BiGRU model architecture

This section presents the ANN architecture of our BiGRU model. The architecture is visualized in Fig. 2.

The architecture comprises bidirectional gated recurrent unit (GRU) layers, as well as fully connected gated layers with various activation functions. Section 4.2 provides details about the BiGRU layers in our architecture. We created two versions of our BiGRU architecture, which we refer to as BiGRU2 and BiGRU4. Both versions differ in the number of BiGRU layers they contain. The numbers in the model names indicate how many BiGRU layers were used

Table 1 The ANN building blocks of our BiGRU occupancy detectors. The table lists two different versions with 2 GRU layers (BiGRU2) and 4 GRU layers (BiGRU4). The particular architecture blocks, the type of the blocks, and the output shapes are listed, where k denotes the number of time steps, n the number of input variables, p the number of individuals to be detected, and d_1, d_2, d_3, d_4 the hidden dimensions of the GRU layers

Model	Block (type)	Input from block	Output shape
BiGRU2	1. Input (InputLayer)		(k, n)
	2. Bidirectional layer (GRU)	1	(k, d_1)
	3. Dropout	2	(k, d_1)
	4. Bidirectional layer (GRU)	3	(d_2)
	5. Hidden layer (dense)	4	n
	6. Hidden layer (dense)	5	k
BiGRU4	7. Output (Softmax)	6	$(p + 1)$
	1. Input (InputLayer)		(k, n)
	2. Bidirectional layer (GRU)	1	(k, d_1)
	3. Dropout	2	(k, d_1)
	4. Bidirectional layer (GRU)	3	(k, d_2)
	5. Dropout	4	(k, d_2)
	6. Bidirectional layer (GRU)	5	(k, d_3)
	7. Dropout	6	(k, d_3)
	8. Bidirectional layer (GRU)	7	(d_4)
	9. Hidden layer (Dense)	8	n
	10. Hidden layer (Dense)	9	k
11. Output (Softmax)	10	$(p + 1)$	

in the respective model instances. Table 1 lists the building blocks of the particular BiGRU models.

The input to the architecture consists of time-series data divided into windows. Each window represents k successive samples, where each sample contains n features. Section 6.2 elaborates on the method we used to extract windows from the datasets. The k successive samples in a window are sequentially processed through the BiGRU layers, resulting in a sequence of states $h_t, h_{t+1}, \dots, h_{t+k}$ for each window. These states are then averaged before the latent vector progresses to the subsequent BiGRU layer. The GRU layers employ Tanh and Sigmoid activation functions for recurrent steps. We introduced dropout layers between the GRU layers with a dropout probability of 30% to aid in regularization. The final BiGRU layer outputs only the last state representation, h_{t+k} , instead of a sequence of states. The dense layers utilize the LeakyReLU [18] activation function. The output layer is a softmax layer with p outputs. In the case of single-occupancy detection, the model is trained using the binary cross-entropy loss function. Conversely, for multi-occupancy detection where the classification problem is not binary, the cross-entropy loss function is implemented for model training. The general ANN architecture design remains similar for both OD tasks. Further details regarding the adaptations made, specific hyper-parameter configurations utilized, and the dataset preprocessing procedures applied will be discussed in Sect. 6.

4.2 Bidirectional gated recurrent unit

This section describes the GRU layers we use in our architecture. We use GRU [19] layers to capture the time-dependency between sequential sensory data. A GRU has an update gate z_t , a reset gate r_t , and a memory \hat{h}_t . The update gate z_t determines how much impact the memory has on the current hidden state h_t , while the reset gate r_t controls the influence of the previous hidden state h_{t-1} that is stored in the memory. According to [20], a GRU can be mathematically expressed with:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$

$$\hat{h}_t = \phi(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h) \tag{3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t, \tag{4}$$

where x_t is the input at time step t , z_t the update gate, r_t the reset gate, \hat{h}_t the memory, h_t the hidden state, σ and ϕ the logistic function and hyperbolic tangent activation functions, and $W_z, W_r, W_h, U_z, U_r, U_h$ the weight matrices, and b_z, b_r, b_h the biases.

The input that is fed into a GRU layer comprises data points of a time series, where the number of time steps considered matches the size of the GRU layer. GRUs have the limitation that time series data is only considered in a certain temporal direction (e.g., the state h_t is computed based on past observations). To overcome this limitation,

we included BiGRU layers in our models. BiGRU layers consist of two separate GRU layers, one processing the input sequence in the forward direction and the other processing it in the reverse direction. This allows our model to capture information from both past and future contexts. The outputs of the forward and backward GRU layers are then combined through averaging to obtain a combined latent state representation.

Traditional RNNs are thrown towards to the challenges of exploding and vanishing gradients, which mitigates their learning capabilities, especially in capturing long-range dependencies [21]. Pascanu et al. [22] highlighted that LSTMs and GRUs were introduced as innovations specifically aimed at mitigating the vanishing gradient problem. These architectures employ gating mechanisms that regulate the flow of information, making them more adept at learning dependencies over longer sequences. However, Rehmer and Kroll [23] showed that the gradient of the GRU is usually smaller than that of the traditional RNN, at least within the constraints of their experimental setup. According to their insights, these smaller gradient magnitudes in GRUs contribute to a smoother loss function. This in effect counteracts against excessively large gradients, providing a more stable training dynamics [23]. The BiGRU models, as an extension of the GRU architecture, inherit these properties. It is imperative to note that while the GRU and consequently BiGRU architectures have mechanisms that make them more robust compared to traditional RNNs in handling long-range dependencies, they do not solve the vanishing or exploding gradient problem, which remain areas of ongoing research and development.

4.3 Complexity analysis

This section compares the space complexity between simple RNN models, our BiGRU2 model, and previously proposed RNN architectures. Understanding the storage requirements of these models is important for assessing their feasibility in real-world applications.

Our BiGRU2 model, an advanced variant of an RNN, incorporates BiGRU layers that allow the model to access both future and past context of the input sequence from its current state [24]. While long short-term memory (LSTM) [25] layers are similar to GRUs, we use GRUs in our models for efficiency reasons. GRUs are computationally more efficient than LSTMs since GRUs have two gates (reset and update gates) [19], whereas LSTMs have three gates (input, output, and forget gates) [25]. GRUs can still efficiently capture dependencies in sequences with only two gates because their reset and update gates work collaboratively to control the flow of information to be remembered or discarded at each time step. According to

[20], the parameter count of a GRU is given by $3(Io + o^2 + o)$, where I is the input dimension and o is the output dimension. This is because there are three sets of operations that require weight matrices of this magnitude. In comparison, LSTMs have $4(Io + o^2 + o)$ parameters [20]. Therefore, GRUs have a lower space complexity than LSTMs. For the experiments presented in this work, we used the Keras API¹ to implement our models, where the total number of parameters for a simple GRU is $3(Io + o^2 + 2o)$, accounting for the inclusion of two separate biases for computational reasons.

Let c denote the context, I the input dimension, d_1, d_2 the hidden dimensions, and o the output dimension. The number of parameters of the BiGRU2 model layers are given by:

1. *First bidirectional layer (GRU)*: $2 \times 3(Id_1 + d_1^2 + 2d_1)$
2. *Second bidirectional layer (GRU)*: $2 \times 3(d_1d_2 + d_2^2 + 2d_2)$
3. *Hidden layer (Dense)*: $d_2 \times I + I$
4. *Hidden layer (Dense)*: $I \times c + c$
5. *Output Layer (Softmax)*: $c \times o + o$

Therefore, the total parameter count of the BiGRU2 model utilized in this work is given by Eq. (5).

$$P(c, I, d_1, d_2, o) = 6(Id_1 + d_1^2 + 2d_1 + d_1d_2 + d_2^2 + 2d_2) + d_2I + I + Ic + c + co + o. \quad (5)$$

Considering the importance of space complexity for practical application of RNN models, we compared the space complexity of the BiGRU2 model against previously proposed RNNs in Table 2. Our BiGRU2 model exhibits a higher space complexity compared to that of a simple BiGRU. However, it is important to note that the hyperparameter configurations of our model were specifically optimized for the multi-occupant detection application and the corresponding evaluation scenarios presented in this work. As shown in Eq. 5, the space complexity of the BiGRU2 model primarily consists of components corresponding to the two bidirectional recurrent layers. Notably, the parameter counts in BiGRU2 have a linear dependence on the context c , as opposed to some of the other models [26–28] where the dependence is quadratic. The linear dependence on the context c can be beneficial in applications where the context varies, as it ensures a more predictable and scalable growth in parameter counts.

The time complexity, commonly measured in floating point operations (FLOPs) or multiply-accumulate operations (MACs), of the models listed in Table 2 heavily depends on various factors, including implementation

¹ https://keras.io/api/layers/recurrent_layers/gru/.

Table 2 Comparison of the space complexity between simple RNNs, previously proposed RNN architectures, and our proposed BiGRU2 model

Work	Space complexity
Simple RNN [20]	$1(Id_1 + d_1^2 + d_1)$ [20]
GRU [19]	$3(Id_1 + d_1^2 + d_1)$ [20]
LSTM [25]	$4(Id_1 + d_1^2 + d_1)$ [20]
Simple BiGRU	$2 \times 3(Id_1 + d_1^2 + d_1)$
MCRN [26]	$cd_1^2 + (c + 1)d_1o + Id_1 + d_1 + I$ [26]
Elman Tower [27]	$cd_1^2 + d_1(I + o) + d_1 + I$ [26]
AR-MCRN-a [28]	$cd_2^2 + Id_2 + 2d_2 + Id_1 + d_1$ [28]
AR-MCRN-b [28]	$cd_2^2 + Id_2 + 2d_2 + d_1d_2 + d_1$ [28]
BiGRU2 (Ours)	Equation (5)

Let I denote the input dimension, d_1, d_2 the hidden dimensions, c the context, and o the output dimension

details, the choice of activation function, the number of features, and the temporal context considered. The specific implementations of the models may differ depending on the deep learning frameworks or libraries utilized, as different optimizations and parallelization techniques are employed to perform computations efficiently. The actual FLOPs required by the models depend heavily on the implementation details and hyperparameter choices. Therefore, in Sect. 6, we report on the actual number of FLOPs required in a forwards pass by our models for each experiment.

5 Datasets

This section presents datasets that contain privacy-aware ambient sensor readings and occupancy annotations, recorded in indoor environments.

5.1 Dataset overview

Many datasets exist that contain privacy-invasive and privacy-aware sensor measurements, whereof some are specifically designed for the development and evaluation of methodologies that should solve the problem of OD. Among the datasets, only very few are appropriate for the development and evaluation of the solution proposed in this work. We consider a dataset to be suitable if it is public, annotated with labels and recorded by privacy-aware monitoring sensors.

Table 3 summarizes datasets that contain privacy-aware ambient sensor readings and occupancy annotations. Table 4 lists additional details about the datasets. We did not consider datasets that were recorded using cameras or

microphones for the development and evaluation of the solution proposed in this work because of their privacy-invasive nature (i.e., they pose a threat to the privacy of individuals).

The office building sensing [5], room occupant estimation [13] and class room occupancy [8] datasets are relevant because they feature ambient characteristics such as CO₂, temperature and humidity sensor readings. However, these datasets are not publically available. The Bluetooth beacons dataset [6] does not contain usual ambient sensor measurements but RSSI recordings. The building fusion set [9] combines ambient information with Wi-Fi data. Both the Bluetooth beacons and the building fusion set are not publically available. The Harvard ODDs [31] dataset contains ambient measurements, including light, temperature, as well as power meter readings and weather data. The Harvard ODDs dataset is publically available. However, it does not contain any labels and therefore cannot be used in supervised learning scenarios. The dataset used in [32] contains state-change sensor measurements and activity labels. The state-change sensors fit into the category of ambient sensors, and were taped onto objects. However, the used sensors are outdated. The UCI occupancy detection [7], Kasteren Ubicomp [29], and CASAS HH101-130 [30] datasets are the most important for the development of the solution proposed in this work and have been utilized in the experiments. These datasets and the novel preprocessing procedure we applied will be presented in Sect. 5.2.

5.2 Utilized datasets and preprocessing

This section presents the datasets we utilized for the development and evaluation of our proposed method and how we preprocessed the particular datasets.

5.2.1 UCI occupancy detection dataset

We consider the UCI occupancy detection [7] dataset to be the most appropriate dataset for human OD experiments. It was created specifically for the development of OD algorithms. The dataset contains recordings from ambient sensors like temperature, humidity, light and CO₂ that monitored a small office room, as well as the timestamp of the recording. Currently, many smart buildings already contain this composition of sensors. The humidity ratio was calculated for every sample. The dataset also contains ground truth binary labels that indicate whether the room is occupied. The researchers generated the ground truth labels automatically with the help of a video surveillance system. However, the binary labels do not indicate how many people were present in the office environment. The number of samples, as well as the overall class distribution of each partition of the dataset, are listed in Table 5. The number

Table 3 Dataset overview with general information. Only the datasets [7, 29, 30] are publically available, annotated with labels and recorded by multiple privacy-aware ambient sensors, which motivated us to use these datasets in our experiments

Work	Dataset	Public	Labels	Sensors type
[7]	UCI occupancy detection	Yes	Yes	Ambient
[29]	Kasteren Ubicomp Dataset	Yes	Yes	Ambient
[30]	CASAS HH 101-130	Yes	Partial	Ambient
[31]	Harvard ODDs	Yes	Partial	Ambient + Power
[32]	Home state change sensors	Yes	Yes	Ambient
[9]	Building fusion set	No	Yes	Ambient + Wi-Fi
[5]	Office building sensing	No	Yes	Ambient
[13]	Room occupant estimation	No	Yes	Ambient
[8]	Class room occupancy	No	Yes	Ambient
[6]	Bluetooth beacons	No	Yes	Bluetooth

Table 4 Dataset overview including the number of subjects, recording duration, temporal resolution and environment

Work	Subjects	Duration	Temporal resolution	Environment
[7]	2	17 days	1 min	Office room
[29]	1	28 days	1 min	Multi-room home
[30]	1–2	few years	dynamic [ms]	Multi-room home
[31]	–	8 months	15 min	Multi-room home
[32]	2	14 days	15 min	Multi-room home
[9]	25	9 days	30/60 s	Office room
[5]	–	55 days	1 / 20 min	Multi-room office
[13]	8	5 days	1 min	Office room
[8]	43	16 days	1 min	Class room
[6]	1	–	–	Multi-room home

Table 5 The overall class distribution of the UCI occupancy detection dataset

Partition	# Samples	Occupied (%)	Not occupied (%)
Training	8143	21	79
Testing 0	2665	36	64
Testing 2	9752	21	79

of samples in the partitions is rather unusual, as the test set contains the majority of all samples. Candanedo et al. [7] also noted that the door of their experimental office room was closed most of the time while the office room was occupied. It's important to note that the state of the door can significantly influence the readings from the ambient sensors installed in the room. This is reflected in the training and testing partition. In the case of testing partition 2, the door was mostly open when someone was in the room. An open door could have influenced the sensor values. However, the authors did not specify the fraction of samples affected. For our experiments, we use the training data only to train our proposed models. The test partitions are only used to test the solution proposed in this work.

Table 6 lists the features contained in the UCI Occupancy Detection dataset [7].

Preprocessing: Following [7], we generated features algorithmically by exploiting the timestamps of the sensor measurements that were recorded in one-minute intervals. Instead of using the raw timestamps, we extracted the number of seconds from midnight for each day and classified the timestamp as either a weekend or a weekday. We did not use the raw timestamps itself. This results in seven features per sample.

Prior to utilizing the samples for model training, we normalized the samples, as explained in Sect. 6.1. Additionally, we grouped successive samples into windows as explained in Sect. 6.2. Subsequently, to prevent potential overfitting, we shuffled the windows post the grouping process.

5.2.2 Kasteren ubicomp dataset

The publicly available Kasteren Ubicomp [29] dataset has been created for the purpose of activity recognition. Although human activities are usually recognized using wearable sensors, this dataset contains ambient sensor readings. Importantly, information about occupancy can be inferred from the activity annotations contained in the

Table 6 The features contained in the UCI Occupancy Detection dataset. The number of seconds since midnight and the differentiation between weekend and weekday are manually engineered features based on the timestamp

Feature	Unit	Example
Timestamp	YYYY-MM-DD hh:mm:ss	2015–02-06 11:05:00
1. ⇒ Sec. since Midnight	Seconds	3600
2. ⇒ Weekend / Weekday	Binary	1
3. Temperature	C°	21,7225
4. Humidity	%	20,7
5. Light	Lux	479
6. CO2	ppm	848,25
7. HumidityRatio	(none)	0,0033

dataset. The dataset is relatively small compared to the other datasets used for the experiments. It has been recorded in a multi-room apartment with a single resident. It contains 1319 sensor firing intervals that have been recorded by digital state-change sensors. The 14 unique sensors and their object placement are listed in Table 7. Besides the sensor firing intervals, the dataset also contains 245 activity intervals that have been manually created with the help of a Bluetooth headset, which was utilized to record verbal annotations. The different activities and their average duration are listed in Table 8. The sensor firings and the corresponding activity annotations were recorded over a time period of 28 days (i.e., 4 weeks from 25.02.2008 to 23.03.2008). The difference in the number of sensor firing intervals and activity intervals indicates that they were recorded independently. Every sample in the dataset comprises a start time, an end time, and an ID. The data formats and examples are listed in Table 9. There are various firing sensors interval lengths ranging from one second to over 24 h. The average firing sensors interval

Table 7 The IDs of the state-change sensors contained in the Kasteren Ubicomp [29] dataset and the objects the sensors were attached to

Sensor ID	Placement
1	Microwave
5	Hall-Toilet door
6	Hall-Bathroom door
7	Cups cupboard
8	Fridge
9	Plates cupboard
12	Front door
13	Dishwasher
14	Toilet flush
17	Freezer
18	Pans cupboard
20	Washingmachine
23	Groceries cupboard
24	Hall-Bedroom door

Table 8 The activity annotations of the Kasteren Ubicomp dataset and the corresponding IDs

Activity ID	Activity	Average duration (min)
1	Leave house	665.7
4	Use toilet	1.8
5	Take shower	9.6
10	Go to bed	485.7
13	Prepare breakfast	3.4
15	Prepare dinner	34.2
17	Get drink	0.9

Table 9 The features contained in the Kasteren Ubicomp dataset

Feature	Format	Example
Start time	DD-Mon-YYYY hh:mm:ss	14-Mar-2008 23:59:37
End time	DD-Mon-YYYY hh:mm:ss	15-Mar-2008 09:53:24
ID	Integer	10

The data formats and examples are listed

length is 44 min, while the average activity length is 171 min. The activities with the longest interval length are 'leave house' and 'go to bed,' with over 11 and over 8 h of duration, respectively. The label 'leave house' (i.e., the activity of a person leaving the home environment) is the most important activity for the experiments conducted in this work because it bears the potential to derive occupancy annotations. The activity 'go to bed' (i.e., the person is sleeping) may cause problems if no sensor firings occur during the activity because an automated OD system may confuse the activity with the absence of the resident.

Preprocessing: We applied two different preprocessing variants to make the dataset suitable for OD experiments. The activity 'leave house' is essential to derive occupancy information, while all other activities indicate the apartment is occupied. The time interval of the activity 'leave house' implies that the resident is absent from the

beginning to the end of the interval. Based on the activity 'leave house,' we extracted 34 absence periods. We used the absence periods to preprocess the dataset in two different ways, as described in the following.

Variant 1: We assigned binary occupancy annotations to the sensor intervals. To determine the label of a sample, we computed the overlap between the sensor firing intervals and the absence periods. We assigned the label 'not occupied,' if there is an overlap of at least 50% between a sensor interval and the absence periods. Otherwise, we assigned the label 'occupied.' In this fashion, we assign a label to every sensor interval. Finally, we extracted six features from each sensor interval: the sensor ID, the length of the interval (i.e., the temporal difference between interval start time and end time), and four additional features. The four additional features are obtained by processing the start time and the end time of the intervals, similar to the UCI Occupancy Detection dataset preprocessing procedure. From each timestamp, the number of seconds since midnight on the same day (NSM), and the week status (WS) indicating a weekend or a weekday, are extracted. All features are normalized as described in Sect. 6.1. Preprocessing variant 1 results in 1319 samples with six features and binary occupancy labels. The class distribution for this preprocessing variant is listed in Table 10. For the experiments we conducted, we used 80% of the sample as the training set and 20% as the testing set and grouped successive samples into windows as explained in Sect. 6.2. Subsequently, to prevent potential overfitting, we shuffled the windows post the grouping process.

Variant 2: Preprocessing variant 2 involves the extraction of equally sized time slices from the sensor intervals, instead of directly treating the particular sensor intervals as samples. This is feasible because date and time are given for each sensor interval. First, the sensor activations are evaluated at a time resolution of one second. For each second, we identify which of the 14 state-change sensors are active. Over the length of the specified time slice, the activations are then averaged, resulting in floating-point values that represent the sensor activations throughout the time slice. This method helps in preserving information regarding sensor activation within each time slice, which is essential as a sensor might be active at the start of a time slice but not active towards the end. The mathematical

representation for the average activation of a sensor over a time slice of length T seconds is given by Equation (6).

$$A_i = \frac{1}{T} \sum_{t=1}^T S_{i,t} \quad (6)$$

In this equation, A_i refers to the average activation of sensor i over the time slice, T represents the length of the time slice in seconds, and $S_{i,t}$ represents the activation state of sensor i at time t (1 if active, 0 otherwise). The dimensionality of the final samples comprises the average sensor activations of all 14 state-change sensors.

The ground truth occupancy labels are again determined by checking whether or not a time slice has at least 50% overlap with any of the absence periods. The resulting class distribution is much more balanced as listed in Table 10. The reason for this is that the non-occupied intervals are generally much longer than the occupied intervals. Therefore, the label 'non-occupied' is assigned to the majority of time slices. In this way, we created samples for time slice lengths of 15/60 s, resulting in 160,015/40,004 samples, respectively. We selected time slice lengths of 15 s and 60 s strategically to accommodate the differing requirements of RNNs and traditional machine learning algorithms that we employed in our study. The 15 s time slice was chosen to facilitate the functioning of RNNs, which are capable at handling sequences of data inputs. By utilizing a 15-second slice, we were able to feed the RNNs with four sequential inputs (amounting to a total of 60 s), thereby enabling the networks to learn and identify dependencies

Table 11 The ambient sensors contained in the CASAS [30] dataset that we included in our experiment

ID	Sensor type
D	Magnetic door sensor
L	Light switch (binary)
LL	Light switch (dimmable)
LS	Light sensor
M	Infrared motion sensor
MA	Wide-area infrared motion sensor
T	Temperature sensor

Table 10 Class distributions of the Kasteren Ubicomp dataset for preprocessing variant 1 (sensor intervals) and variant 2 (time slice length 15/60 s) with binary occupancy labels

Variant	# Samples	# Features	Occupied (%)	Not occupied (%)
1	1319	6	83.40	16.60
2	160,015/40,004	14	43.59	56.41

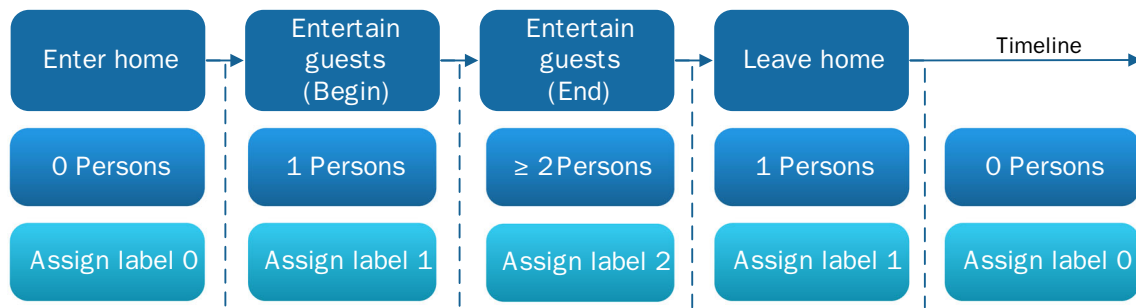


Fig. 3 The labeling procedure we apply to the CASAS dataset. We differentiate three different cases of occupancy so that we can not only use the dataset for occupancy detection but also for multi-occupancy detection

Table 12 Class distributions of the CASAS HH111 and HH112 datasets. The number of samples refers to time slice length 15/60 s

Apartment	# Samples	# Features	Occupied (%)	Not occupied (%)
HH111	351,217/87,805	62	60,82	39,18
HH112	622,076/155,520	45	64,95	35,05

across a short time series of data points. Conversely, the 60 s time slice was designated to suit the input requirements of traditional machine learning algorithms utilized in our analysis. By adopting these specified time slice lengths, we aimed to tailor the input data optimally for both types of algorithms, thereby facilitating a more comparative and comprehensive analysis of their performance. We used 70% of the sample as the training set and 30% as the testing set and grouped successive samples into windows as explained in Sect. 6.2. Subsequently, to prevent potential overfitting, we shuffled the windows post the grouping process.

5.2.3 CASAS HH101-HH130 datasets

The CASAS HH101-HH130 [30] datasets comprise a collection of different datasets that were created at the Washington State University (WSU). WSU sent a selection of smart ambient sensors to volunteers, who installed the sensors in their apartments. The identifiers HH101-HH130 refer to 30 apartments that are mostly different in structure, only a few of the apartments are identical. All recordings originate from single resident home environments, except the HH107 and HH121 apartments are two resident apartments. The apartments have different layouts, so that the ambient sensors are tailored to them (i.e., the numbers of sensors differ). The ambient sensors used for the measurements are identical, but the locations and number of sensors vary depending on the apartments. The selection of sensors includes light, motion, magnetic and temperature sensors. The sensors we included in our experiments are listed in Table 11.

The number of sensor measurements ranges from under 200 thousand to over 16 million, depending on the apartment. The average recording period is 951 days. However, only 4,63% of all measurements have activity annotations. The apartments have different numbers of sensors. Thus, the dimensionality of the samples (i.e., the number of features) varies between the particular apartments. The annotations mostly indicate the start or end of an activity, thus we extracted activity intervals, similarly to the pre-processing procedure of the Kasteren Ubicomp dataset.

Preprocessing: The most crucial activity annotations are 'Leave home' and 'Enter home' because we use these annotations to derive occupancy and absence intervals. In

Table 13 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing MLP model

Block (type)	Output Shape	Parameters
1. Input (InputLayer)	(7)	0
2. Hidden Layer (Dense)	(21)	168
3. Dropout	(21)	0
4. Hidden Layer (Dense)	(21)	462
4. Hidden Layer (Dense)	(7)	154
5. Hidden Layer (Dense)	(4)	32
7. Output (Softmax)	(2)	10
Total number of parameters		826
Model size		3,25 KiB
FLOPs (forwards pass)		1607

* The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

addition, we extracted intervals in which guests are present in the apartments. Guest intervals always occur outside absence intervals. We used the datasets recorded in the HH112 and HH111 apartments for occupancy detection and multi-occupancy detection, respectively. Figure 3 shows the labeling procedure we applied to the dataset. In the case of multi-occupancy detection, we used the activity annotation 'Entertain guests' to derive the presence of multiple individuals in an apartment. The state 'more than one occupant' was observed in less than 7% of the intervals.

The number of guests is not specified in the CASAS [30] dataset. Therefore, it is not possible to derive the number of occupants using the dataset. Although it is possible to derive whether one or multiple persons are present in intervals with guests. We differentiate between the following occupancy states: (a) no occupant, (b) exactly one occupant, (c) more than one occupant.

Table 14 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing BiGRU models

Model	Block (type)	Output shape	Parameters
BiGRU2	1. Input (InputLayer)	(1, 7)	0
	2. Bidirectional layer (GRU)	(1, 28)	6216
	3. Dropout	(1, 28)	0
	4. Bidirectional layer (GRU)	(28)	9744
	5. Hidden layer (Dense)	(7)	203
	6. Hidden layer (Dense)	(1)	8
	7. Output (Softmax)	(2)	4
	Total number of parameters		16.175
	* Model size		63,18 KiB
	FLOPs (forward pass)		32,630
BiGRU4	1. Input (InputLayer)	(1, 7)	0
	2. Bidirectional layer (GRU)	(1, 28)	6216
	3. Dropout	(1, 28)	0
	4. Bidirectional layer (GRU)	(1, 28)	9744
	5. Dropout	(1, 28)	0
	6. Bidirectional layer (GRU)	(1, 14)	3696
	7. Dropout	(1, 14)	0
	8. Bidirectional layer (GRU)	(10)	1560
	9. Hidden layer (Dense)	(7)	77
	10. Hidden layer (Dense)	(1)	8
	11. Output (Softmax)	(2)	4
Total number of parameters		21.305	
* Model size		83,22 KiB	
FLOPs (forward pass)		42,950	

* V1 and V2 denote annotation variants 1 and 2, respectively

** The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

Similarly to the preprocessing procedure of the Kasteren Ubicomp dataset, we split the datasets into equally sized time slices with a length of 15/60 s and annotated them using the occupancy intervals. Table 12 lists the resulting number of samples after preprocessing and the class distribution for both apartments, respectively.

We grouped successive samples into windows as explained in Sect. 6.2 and shuffled the windows post the grouping process, to prevent potential overfitting.

6 Experimental setup

This section presents the general preprocessing procedures that all datasets we utilized have in common and the setups for the experiments conducted.

6.1 Feature normalization

The features (i.e., sensor measurements) contained in the datasets vary significantly in terms of value ranges. To facilitate a smoother gradient descent flow and prevent a bias towards features with higher magnitude values, we scale these features. The Tanh activation function utilized in our ANN architectures is zero-centered; hence, we scale the values to a range of [0, 1] using Min–Max normalization [33]. Let x represent a value of an input feature, and x_{max} and x_{min} represent the maximum and minimum value of this feature, respectively. The normalized value of x , represented as $n(x)$, is given by:

$$n(x) = \frac{x - x_{min}}{x_{max} - x_{min}}. \quad (7)$$

6.2 Window extraction

In this section, we describe the process of converting time-series data into windows, facilitating data processing through our proposed ANN architecture. It is critical to

Table 15 The hyper-parameter configuration we used to train our ANN models

Hyper-parameter	Value
Learning rate	$2e - 3$
Learning decay β_1	0.9
Learning decay β_2	0.999
Batch size	100
Dropout probability	30%
Training epochs	12
Window size k	1 (i.e., 1 min)

Table 16 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing BiGRU models

Model	Block (type)	Output shape *V1	Params. *V1	Output shape *V2	Params. *V2
BiGRU2	1. Input (InputLayer)	(4, 6)	0	(4, 14)	0
	2. Bidirectional Layer (GRU)	(4, 40)	11.520	(4, 20)	4320
	3. Dropout	(4, 40)	0	(4, 20)	0
	4. Bidirectional Layer (GRU)	(5)	1410	(5)	810
	5. Hidden Layer (Dense)	(6)	36	(14)	84
	6. Hidden Layer (Dense)	(4)	28	(4)	60
	7. Output (Softmax)	(2)	10	(2)	10
	Total number of parameters		13.004		5284
	** Model size		50,80 KiB		20,64 KiB
	FLOPs (forward pass)		26,196		10,668
BiGRU4	1. Input (InputLayer)	(4, 6)	0	(4, 14)	0
	2. Bidirectional Layer (GRU)	(4, 40)	11.520	(4, 20)	4320
	3. Dropout	(4, 40)	0	(4, 20)	0
	4. Bidirectional Layer (GRU)	(4, 30)	12.960	(4, 20)	5040
	5. Dropout	(4, 30)	0	(4, 20)	0
	6. Bidirectional Layer (GRU)	(4, 20)	6240	(4, 10)	1920
	7. Dropout	(4, 20)	0	(4, 10)	0
	8. Bidirectional Layer (GRU)	(5)	810	(5)	510
	9. Hidden Layer (Dense)	(6)	36	(14)	84
	10. Hidden Layer (Dense)	(4)	28	(4)	60
	11. Output (Softmax)	(2)	10	(2)	10
	Total number of parameters		31.604		11.944
	** Model size		123,45 KiB		46,66 KiB
FLOPs (forward pass)		63,596		24,108	

* V1 and V2 denote annotation variants 1 and 2, respectively

** The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

Table 17 The hyper-parameter configuration we used to train our ANN models

Hyper-parameter	Variant 1	Variant 2
Learning rate	$2e - 3$	$2e - 3$
Learning decay β_1	0.9	0.9
Learning decay β_2	0.999	0.999
Batch size	50	400
Dropout probability	30%	30%
Training epochs	250	25
Window size k	4	4

retain the temporal dependency observed between sequential data points. Consequently, we transformed the time-series data into individual feature vectors, or windows, employing the sliding window method [34]. A window is defined as a tuple denoted by $W_i \equiv \langle s_i, s_{i+1}, s_{i+k-1} \rangle$, where s_i refers to the i -th sample and k represents the window size. During the inference phase,

our ANN structure categorizes each window individually. This process involves sliding a fixed-size window across the entire time series, extracting $n - k + 1$ windows represented by $W_1, W_2, \dots, W_{n-k+1}$, where n indicates the total sample count within a dataset. In the training phase, all windows are extracted from the training data for model training. The ground truth label corresponding to a window is determined by the label of its final contained sample. In the testing phase, all windows from the test data are extracted and utilized for model evaluation. This process ensures that the initial verdict from our occupancy detector is obtained after observing k successive samples.

6.3 Experiment 1: UCI occupancy detection dataset

In this experiment, we evaluate OD using the UCI occupancy detection dataset [7]. We compared the performance of various traditional algorithms with RNN variants, including our BiGRU occupancy detector. The traditional algorithms examined are SVM, quadratic discriminant

Table 18 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing MLP models

Block (type)	Output shape *V1	Params. *V1	Output shape *V2	Params. *V2
1. Input (InputLayer)	(6)	0	(14)	0
2. Hidden Layer (Dense)	(40)	280	(20)	300
3. Dropout	(40)	0	(20)	0
4. Hidden Layer (Dense)	(5)	205	(5)	105
4. Hidden Layer (Dense)	(6)	36	(14)	84
5. Hidden Layer (Dense)	(4)	28	(4)	60
7. Output (Softmax)	(2)	10	(2)	10
Total number of parameters		559		559
* Model size		2,18 KiB		2,18 KiB
FLOPs (forward pass)		1071		1083

*V1, V2 denote annotation variants 1 and 2, respectively

** The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

Table 19 The hyper-parameter configuration we used to train our ANN models

Hyper-parameter	Value
Learning rate	$2e - 3$
Learning decay β_1	0.9
Learning decay β_2	0.999
Batch size	250
Dropout probability	30%
Training epochs	25
Window size k	4

analysis (QDA), and k-NN. The hyperparameters for these algorithms are detailed below:

- SVM (all variants): Shrinking heuristic enabled, $1e - 3$ tolerance for stopping criterion, and one-vs-rest decision function.
 - SVM-lin: Linear kernel.
 - SVM-poly: 3rd-degree polynomial kernel, scale gamma.
 - SVM-rbf: radial basis function (RBF) kernel, scale gamma.
- k-NN (all variants): Uniform weights, automatic algorithm determination, leaf size of 30, and Euclidean distance metric.
 - 4-NN: $k = 4$.
 - 20-NN: $k = 20$.
- QDA: Rank estimation threshold set at $1e - 4$.

Table 13 lists the ANN building blocks of our MLP. The dense layers of our MLP use the Leaky ReLU activation function with $a = 0.3$. We included a dropout layer with a 50% probability to mitigate overfitting. In this binary classification scenario, the output layer is of size 2 and uses the Softmax activation function. We trained the MLP for

10 epochs with the Adam optimizer [35], a learning rate of $1e - 3$, and a Batch size of 50 samples.

Besides the traditional algorithms, we also created instances of RNNs, including our BiGRU models for this experiment. Layer configurations were optimized through hyperparameter tuning. The architecture, parameter details, and computational requirements of our highest-performing BiGRU models are listed in Table 14. The number of model parameters greatly depends on the number of input features n and the window size k . In this experiment, we set the window size to $k = 1$ because we want to compare the performance of our RNN models directly to the detection performance reported in [7]. Although our RNN models cannot unfold their true potential, when the inputs only consist of single samples. We used the Nadam [36] optimizer for ANN parameter optimization. Table 15 lists additional hyper-parameters we used to train our ANN models in this experiment.

6.4 Experiment 2: Kasteren ubicomp dataset

This experiment investigates OD utilizing the Kasteren Ubicomp [29] dataset. We employed two distinct preprocessing variants on the dataset, detailed in Sect. 5.2.2. Analogous to Experiment 1, we assessed the performance of various traditional algorithms, including SVM, k-NN, and QDA, comparing them with our ANNs. Furthermore, we compare with the LDA and classification and regression tree (CART) algorithms. The hyperparameters defined for the LDA and CART algorithms are as follows:

- LDA linear discriminant analysis, implemented using singular value decomposition with a rank estimation threshold set at $1e - 4$.
- CART A decision tree classifier employing the Gini index as the split criterion and selecting the optimal split at each decision node. The minimum numbers of

samples required to split a node and to be at a leaf node are set to 2 and 1, respectively.

We created instances of both BiGRU versions, BiGRU2 and BiGRU4, for this experiment. The configurations of the top-performing instances are listed in Table 16. Depending on the preprocessing variant applied to the dataset, slight adjustments were made to the GRU layer sizes. Details on the hyperparameter configurations used to train our ANN models are listed in Table 17. Similarly, the architecture details of the MLPs we created are summarized in Table 18. Additional details regarding the experimental setups for both preprocessing variants are provided below:

Variant 1: The recurrent models utilized a window size of $k = 4$ samples. In contrast, all other algorithms, including our MLP, were configured with a window size of $k = 1$. Each sample has $n = 6$ features. Each sample comprised $n = 6$ features. The MLP training was conducted over 250 epochs, employing the adam optimizer with a learning rate of $1e - 3$.

Variant 2: The recurrent models utilized a window size of $k = 4$ samples, where each sample corresponds to a 15-second time slice. In comparison, all other algorithms, including our MLP, were set with a window size of $k = 1$

sample, representing a 60-second time slice. Each sample in this variant incorporated $n = 14$ features. The MLP was trained over 10 epochs using the Adam optimizer with a learning rate of $2e - 3$.

6.5 Experiment 3: CASAS HH112 dataset

This experiment investigates OD using the CASAS HH112 [30] dataset. Table 20 lists the architecture details of our top-performing BiGRU models utilized for this experiment. The hyper-parameters utilized to train our ANN models are listed in Table 19. The hyper-parameters are mostly consistent with those used in the other experiments.

6.6 Experiment 4: CASAS HH111 dataset

This experiment investigates multi-occupancy detection utilizing the CASAS HH111 [30] dataset. Table 21 lists the architecture details of our top-performing BiGRU models employed in this experiment. Most of the other hyper-parameters are consistent with those used in Sect. 6.5, with a notable alteration in the output shape of the Softmax layer,

Table 20 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing BiGRU models

Model	Block (type)	Output shape	Parameters
BiGRU2	1. Input (InputLayer)	(4, 45)	0
	2. Bidirectional Layer (GRU)	(4, 50)	29.100
	3. Dropout	(4, 50)	0
	4. Bidirectional Layer (GRU)	(10)	3720
	5. Hidden Layer (Dense)	(45)	495
	6. Hidden Layer (Dense)	(4)	184
	7. Output (Softmax)	(2)	10
	Total number of parameters		33.509
	* Model size		130,89 KiB
	FLOPs (forward pass)		67.237
BiGRU4	1. Input (InputLayer)	(4, 45)	0
	2. Bidirectional Layer (GRU)	(4, 50)	29.100
	3. Dropout	(4, 50)	0
	4. Bidirectional Layer (GRU)	(4, 80)	22.080
	5. Dropout	(4, 80)	0
	6. Bidirectional Layer (GRU)	(4, 60)	20.160
	7. Dropout	(4, 60)	0
	8. Bidirectional Layer (GRU)	(10)	4320
	9. Hidden Layer (Dense)	(45)	495
	10. Hidden Layer (Dense)	(4)	184
	11. Output (Softmax)	(2)	10
Total number of parameters		76.349	
* Model size		298,24 KiB	
FLOPs (forward pass)		297.957	

* The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

Table 21 Specifications of the architecture building blocks, model parameters, output shapes, and FLOPs for the top-performing BiGRU models

Model	Block (type)	Output shape	Parameters
BiGRU2	1. Input (InputLayer)	(4, 62)	0
	2. Bidirectional layer (GRU)	(4, 50)	34.200
	3. Dropout	(4, 50)	0
	4. Bidirectional layer (GRU)	(10)	3720
	5. Hidden layer (Dense)	(62)	682
	6. Hidden layer (Dense)	(4)	252
	7. Output (Softmax)	(3)	15
	Total number of parameters		38.869
	* Model size		151,81 KiB
	FLOPs (forward pass)		77.930
	BiGRU4	1. Input (InputLayer)	(4, 62)
2. Bidirectional layer (GRU)		(4, 50)	34.200
3. Dropout		(4, 50)	0
4. Bidirectional layer (GRU)		(4, 80)	63.360
5. Dropout		(4, 80)	0
6. Bidirectional layer (GRU)		(4, 60)	51.120
7. Dropout		(4, 60)	0
8. Bidirectional layer (GRU)		(10)	4320
9. Hidden layer (Dense)		(62)	682
10. Hidden layer (Dense)		(4)	252
11. Output (Softmax)		(3)	15
Total number of parameters		153.949	
* Model size		601,36 KiB	
FLOPs (forward pass)		308.664	

* The model size is represented in kibibytes (KiB), calculated based on the Float32 data type

which has been adjusted to 3 to distinguish between three classes in this experiment.

6.7 Evaluation metrics

In evaluating the performance of our models, we employ several commonly used performance metrics, including accuracy, F1-score, precision, and recall.

7 Results

This section presents the results of the particular experiments. The performance of our BiGRU solution is compared to several state-of-the-art algorithms.

7.1 Experiment 1: UCI occupancy detection dataset

The results of this experiment are listed in Table 22. In [7], the authors analyzed the performance of the RF, GBM, CART and LDA algorithms. To facilitate a direct comparison, we set the windows size necessary to train and test

our RNN models to $k = 1$ samples, aligning with the results reported in [7]. However, this setting prevents our RNN models to unfold their full potential as they do not profit from patterns extracted between successive time steps. In addition to our BiGRU models, we trained QDA, k-NN, and SVM models, along with our ANN implementations on the respective dataset. The SVM models yield accuracies of 97.90%, 97.86%, and 97.94% on test set 1, with the SVM-rbf model showing the highest accuracy reported for this test set in comparison to previous work.

Our MLP, BiGRU2, BiGRU4 models yielded accuracies of up to 97.86%, 97.71%, 97.60% on test set 1, marking a noticeable improvement over prior work, but without setting new standards. In particular, our BiGRU2 model and MLP surpassed other solutions on test set 2, with the MLP even slightly outperforming our BiGRU models by margins of 0.15 and 0.05% points on test sets 1 and 2, respectively. Additionally, we incorporated well-established RNN architectures such as GRU and LSTM, enhancing the depth of our comparison. However, these showed lower performance compared to our BiGRU models on test set 2. Overall, the ANN models proposed in this work outperform the state-of-the-art on test set 2.

Table 22 Comparison to previous work using the UCI occupancy detection dataset. The performance metrics reported in previous work are limited, only the accuracy has been reported. For the ANN

models, average and top performances are significantly different. Our BiGRU2 and MLP model outperforms the state-of-the-art

Model	Test 1 Avg Accuracy (%)	Test 2 Avg Accuracy (%)	Test 1 Top Accuracy (%)	Test 2 Top Accuracy (%)
RF [7]	N/A	N/A	95,53	98,06
GBM [7]	N/A	N/A	95,76	96,10
CART [7]	N/A	N/A	94,52	96,52
LDA [7]	N/A	N/A	97,90	99,33
SVM [10]	N/A	N/A	97,90	N/A
K-NN [10]	N/A	N/A	95,90	N/A
RBFN [10]	N/A	N/A	97,00	N/A
MCRN [10]	N/A	N/A	97,60/93,20	N/A
MVCNN [11]	N/A	N/A	97,40	97,72
DDQN-PER [12]	N/A	N/A	96,40	98,20
QDA	N/A	N/A	97,67	99,02
4-NN	97,00	94,29	97,00	94,29
20-NN	97,34	94,16	97,34	94,16
SVM-lin	97,90	98,95	97,90	98,95
SVM-poly	97,86	97,64	97,86	97,64
SVM-rbf	97,94	98,53	97,94	99,46
MLP	97,49	98,66	97,86	99,44
GRU	N/A	N/A	97,86	96,86
LSTM	N/A	N/A	97,71	95,93
BiGRU2	97,22	96,92	97,71	99,39
BiGRU4	97,24	97,83	97,60	98,55

Table 23 Comparison to previous work on occupancy detection, variant 1 (sensor intervals) using the Kasteren Ubicomp dataset

Model	Accuracy (%)	F1-Score (%)	Precision (%)	Recall (%)
4-NN	95,85	97,48	100,00	95,09
QDA	86,04	92,31	86,38	99,11
LDA	84,91	91,74	85,38	99,11
CART	96,60	97,98	98,64	97,32
SVM-rbf	85,66	92,18	85,50	100,00
MLP	87,55	92,48	89,04	96,21
GRU	89,77	93,79	91,89	95,77
LSTM	91,29	94,69	93,18	96,24
BiGRU2	91,67	94,91	93,61	96,24
BiGRU4	85,23	91,39	86,25	97,18

7.2 Experiment 2: kasteren ubicomp dataset

In this experiment, the evaluations were conducted on two preprocessing variants, each differing significantly in terms of occupancy annotations and class distribution. The results for each variant are listed in Tables 23 and 24.

Variant 1: In this variant, the CART model was the top-performing instance, with an F1-score of 97.98%. Despite the k-NN algorithm yielded an 97.48% F1-score. Among the ANN models, our BiGRU2 was the top-performing

instance with an F1-score of 94.91%. Notably, with 97.18%, our BiGRU4 model yielded the highest recall rate among the ANN models.

Variant 2: Analyzing Table 24, it is clear that the ANN models demonstrated quite similar performance across various metrics. Focusing on the F1-score, the MLP, GRU, LSTM, BiGRU2, and BiGRU4 models all recorded scores within a narrow range, oscillating between 77.49% and 77.69% F1-score. This similarity is mirrored in their accuracy, precision, and recall metrics as well, indicating a

Table 24 Comparison to previous work on occupancy detection, variant 2 (60 s slices for traditional algorithms, 4x15 s slices for RNN

models) using the Kasteren Ubicomp dataset. The ANN models achieved a consistent performance

Model	Accuracy (%)	F1-Score (%)	Precision (%)	Recall (%)
4-NN	81,29	78,01	79,38	76,70
QDA	61,26	19,61	96,18	10,92
LDA	75,76	69,20	76,86	62,92
CART	81,29	78,02	79,36	76,73
SVM-rbf	80,89	77,41	79,27	75,63
MLP	81,39	77,69	80,23	75,31
GRU	80,90	77,65	80,12	75,32
LSTM	80,80	77,50	80,09	75,06
BiGRU2	80,88	77,62	80,13	75,27
BiGRU4	80,79	77,49	80,08	75,05

Table 25 Comparison to previous work on occupancy detection (60 s slices for traditional algorithms, 4x15 s slices for RNN models) using

the CASAS HH112 dataset. Our BiGRU4 model achieved the highest recall rate compared to the other RNN models

Model	Accuracy (%)	F1-Score (%)	Precision (%)	Recall (%)
4-NN	90,76	92,81	93,92	91,72
QDA	64,97	78,77	64,97	100,00
LDA	69,72	78,03	73,80	82,78
CART	92,97	94,72	92,38	97,19
SVM-rbf	72,87	79,28	78,67	79,89
MLP	75,61	81,97	78,44	85,82
GRU	87,51	90,56	88,81	92,38
LSTM	86,05	89,53	87,17	92,03
BiGRU2	88,27	91,12	89,43	92,88
BiGRU4	87,29	90,32	89,19	91,49

consistent performance across all these ANN models. In the case of the traditional algorithms, the CART algorithm performed best, yielding an F1-score of 78.02%. The QDA and LDA algorithms were observed to have underperformed when compared to the other models. The QDA model, in particular, displayed a noticeably lower performance, with an F1-score of 19.61%. While the ANN models exhibited a homogeneous performance, the CART model displayed a marginally better output, demonstrating its efficacy in the task.

7.3 Experiment 3: CASAS HH112 dataset

In this experiment, we analyzed the performance tested on the CASAS HH112 dataset as presented in Table 25. The CART model proved to be the top-performing instance with regard to both accuracy and F1-score, which were 92.97% and 94.72%, respectively. The worst performing instance, when considering the F1-score as primary indicator of performance, was the LDA algorithm. It yielded

the lowest F1-score at 78.03%. The RNN models (GRU, LSTM, BiGRU2, and BiGRU4) yielded close performance, especially when focusing on the F1-scores which were all within the range of 89.53 to 91.12%. The BiGRU2 model was the top-performing instance among the RNN models.

7.4 Experiment 4: CASAS HH111 dataset

This section presents the results of our multi-occupancy detection experiment, which are listed in Table 26. Among the traditional algorithms, 4-NN demonstrated a good performance, with an F1-Score of 93.96%. The CART model was the top-performing instance across all metrics, achieving the highest F1-Score of 96.16%. The QDA model struggled significantly in this experiment, managing only a 31.64% F1-Score. Among the ANN models, the MLP model showed the lowest performance, with an F1-score of 76.93%. The RNN models present a noticeable improvement, with F1-scores above 90%. They maintained balance between precision and recall, with the LSTM

Table 26 Comparison to previous work on multi-occupancy detection (60 s slices for traditional algorithms, 4x15 s slices for RNN models) using the CASAS HH111 dataset

Model	Accuracy (%)	F1-Score (%)	Precision (%)	Recall (%)
4-NN	94,11	93,96	94,58	93,46
QDA	45,53	31,64	25,71	56,32
LDA	73,20	69,96	74,61	66,95
CART	96,09	96,16	96,69	95,66
SVM-rbf	76,87	76,05	79,87	73,27
MLP	77,64	76,93	80,79	74,30
GRU	91,67	92,45	93,10	91,88
LSTM	92,18	92,94	93,50	92,42
BiGRU2	89,44	90,41	91,50	89,43
BiGRU4	90,82	91,55	92,32	90,84

slightly outperforming the GRU and BiGRU models by 0.49 and 1.39% points F1-score, respectively.

8 Conclusion

In this work, we presented occupancy and multi-occupancy detection experiments. The strength of our proposed BiGRU occupancy detector lies in its ability to harness both historical and future contexts through bidirectional recurrent layers, making it adept at identifying complex patterns within sensory data. Besides the BiGRU models, a major contribution of this work is the sophisticated pre-processing procedure we employed in preparing the Kasteren UbiComp and CASAS dataset. This procedure not only annotated the CASAS dataset with occupancy labels but also tailored it to address multi-occupancy detection experiments. Our preprocessing contributed significantly, as the Kasteren UbiComp and CASAS dataset were originally developed for activity recognition rather than OD. Our evaluations were performed with three different datasets recorded exclusively with privacy-aware ambient sensors. Notably, with accuracies and F1-scores exceeding 90%, our models not only excelled in single-occupancy detection but also showed great promise in the more challenging task of multi-occupancy detection. In addition, our work presented insightful comparisons between the BiGRU models and other state-of-the-art RNN algorithms. Moreover, our ANN models outperformed previous work on the UCI Occupancy Detection dataset with accuracies up to 99.44%. This work lays the foundation for further research and innovation in the field of occupancy detection.

Acknowledgements This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

Funding Open Access funding enabled and organized by Projekt DEAL. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Availability of data and materials The UCI Occupancy Detection dataset analyzed during the current study is available in the UCI machine learning repository [37]. The Kasteren UbiComp dataset analyzed during this study is included in the supplementary information files of the original published article [29]. The CASAS dataset analyzed during the current study is available in the CASAS repository of the Washington State University [38].

References

- Chen Z, Jiang C, Xie L (2018) Building occupancy estimation and detection: a review. *Energy Build* 169:260–270. <https://doi.org/10.1016/j.enbuild.2018.03.084>
- Narayanaswamy B, Balaji B, Gupta R, Agarwal Y (2014) Data driven investigation of faults in HVAC systems with model, cluster and compare (MCC). In: Proceedings of the 1st ACM conference on embedded systems for energy-efficient buildings. *BuildSys '14*, pp 50–59. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/2674061.2674067>
- Mills E (2011) Building commissioning: a golden opportunity for reducing energy costs and greenhouse gas emissions in the united states. *Energy Eff* 4(2):145–173
- Black J, Velastin SA, Boghossian BA (2005) A real time surveillance system for metropolitan railways. In: *Advanced*

- video and signal based surveillance, 2005 IEEE international conference on video and signal based surveillance (AVSS'05), 15–16 September 2005, Como, Italy, pp 189–194. IEEE Computer Society, Washington, DC, USA. <https://doi.org/10.1109/AVSS.2005.1577265>
5. Dong B, Andrews B, Lam KP, Höyneck M, Zhang R, Chiou Y-S, Benitez D (2010) An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network. *Energy Build* 42(7):1038–1046
 6. Alhamoud A, Nair AA, Gottron C, Böhnstedt D, Steinmetz R (2014) Presence detection, identification and tracking in smart homes utilizing bluetooth enabled smartphones. In: IEEE 39th conference on local computer networks, Edmonton, AB, Canada, 8–11 September, 2014 - workshop proceedings, pp 784–789. IEEE Computer Society, Washington, DC, USA. <https://doi.org/10.1109/LCNW.2014.6927735>
 7. Candanedo LM, Feldheim V (2016) Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy Build* 112:28–39
 8. Szczurek A, Maciejewska M, Pietrucha T (2017) Occupancy determination based on time series of co2 concentration, temperature and relative humidity. *Energy Build* 147:142–154
 9. Wang W, Chen J, Hong T (2018) Occupancy prediction through machine learning and data fusion of environmental sensing and wi-fi sensing in buildings. *Autom Constr* 94:233–243
 10. Hoori AO, Motai Y (2018) Multicolumn RBF network. *IEEE Trans Neural Netw Learn Syst* 29(4):766–778. <https://doi.org/10.1109/TNNLS.2017.2650865>
 11. Liu C, Hsaio W, Tu Y (2019) Time series classification with multivariate convolutional neural network. *IEEE Trans Ind Electron* 66(6):4788–4797. <https://doi.org/10.1109/TIE.2018.2864702>
 12. Fähmann D, Jorek N, Damer N, Kirchbuchner F, Kuijper A (2022) Double deep q-learning with prioritized experience replay for anomaly detection in smart environments. *IEEE Access* 10:60836–60848. <https://doi.org/10.1109/ACCESS.2022.3179720>
 13. Han H, Jang K-J, Han C, Lee J (2013) Occupancy estimation based on co2 concentration using dynamic neural network model. *Proc. AIVC* 13
 14. Jin M, Bekiaris-Liberis N, Weekly K, Spanos C, Bayen A (2015) Sensing by proxy: occupancy detection based on indoor co2 concentration. *UBICOMM* 2015:14
 15. Xia L, Chen C, Aggarwal JK (2011) Human detection using depth information by Kinect. In: IEEE conference on computer vision and pattern recognition, CVPR workshops 2011, Colorado Springs, CO, USA, 20–25 June, 2011, pp 15–22. IEEE Computer Society, Washington, DC, USA. <https://doi.org/10.1109/CVPRW.2011.5981811>
 16. Corvee, E, Bak, S, Brémond F (2012) People detection and re-identification for multi surveillance cameras. In: Proceedings of the international conference on computer vision theory and applications, Volume 2: VISAPP, (VISIGRAPP 2012), pp 82–88. SciTePress, Setubal, Portugal. <https://doi.org/10.5220/0003808600820088>. INSTICC
 17. Pirttikangas S, Tobe Y, Thepvilajanapong N (2010) Smart environments for occupancy sensing and services. In: Nakashima H, Aghajan HK, Augusto JC (eds) Handbook of ambient intelligence and smart environments, pp 825–849. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-93808-0_31
 18. Maas AL, Hannun AY, Ng AY, et al. (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proc. Icml, vol 30, p 3 (2013). Atlanta, Georgia, USA
 19. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1724–1734. Association for Computational Linguistics, Doha, Qatar. <https://doi.org/10.3115/v1/D14-1179>. <https://aclanthology.org/D14-1179>
 20. Dey R, Salem FM (2017) Gate-variants of gated recurrent unit (GRU) neural networks. In: IEEE 60th international midwest symposium on circuits and systems, MWSCAS 2017, Boston, MA, USA, August 6–9, 2017, pp 1597–1600. IEEE, Washington, DC, USA. <https://doi.org/10.1109/MWSCAS.2017.8053243>
 21. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Netw* 5(2):157–166
 22. Pascanu R, Mikolov T, Bengio Y (2012) Understanding the exploding gradient problem. *CoRR* [arXiv:1211.5063](https://arxiv.org/abs/1211.5063)
 23. Rehmer A, Kroll A (2020) On the vanishing and exploding gradient problem in gated recurrent units. *IFAC-PapersOnLine* 53(2):1243–1248
 24. Salehinejad H, Baarbe J, Sankar S, Barfett J, Colak E, Valaee S (2018) Recent advances in recurrent neural networks. *CoRR* [arXiv:1801.01078](https://arxiv.org/abs/1801.01078)
 25. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
 26. Huang B, Rashid T, Kechadi M (2007) Multi-context recurrent neural network for time series applications. *Int J Comput Inf Eng* 1(10):3086–3095
 27. Elman JL (1990) Finding structure in time. *Cogn Sci* 14(2):179–211. https://doi.org/10.1207/s15516709cog1402_1
 28. Rashid T, Huang B, Kechadi M, Gleeson B (2006) Auto-regressive recurrent neural network approach for electricity load forecasting. *Int J Comput Intell* 3(1):1–9
 29. van Kasteren T, Noulas A, Englebienne G, Kröse B (2008) Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on ubiquitous computing. UbiComp '08, pp 1–9. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/1409635.1409637>
 30. Cook DJ, Crandall AS, Thomas BL, Krishnan NC (2013) CASAS: a smart home in a box. *Computer* 46(7):62–69. <https://doi.org/10.1109/MC.2012.328>
 31. Makonin S (2015) ODDs: occupancy detection dataset. *Harv Dataverse*. <https://doi.org/10.7910/DVN/2K9FFE>
 32. Tapia EM, Intille SS, Larson K (2004) Activity recognition in the home using simple and ubiquitous sensors. In: Ferscha A, Mattern F (eds) Pervasive computing. Springer, Berlin, Heidelberg, pp 158–175
 33. Han J, Kamber M, Pei J (2011) Data transformation and data discretization. *Data mining: concepts and techniques*, pp 111–118
 34. Dietterich TG (2002) Machine learning for sequential data: a review. In: Caelli T, Amin A, Duin RPW, de Ridder D, Kamel M (eds) Structural, syntactic, and statistical pattern recognition. Springer, Berlin, pp 15–30
 35. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
 36. Dozat, T.: Incorporating nesterov momentum into adam (2016)
 37. Candanedo L (2016) UCI Machine Learning Repository. <https://archive.ics.uci.edu/dataset/357/occupancy+detection>. Accessed: 24 Sep 2023
 38. Cook DJ, Crandall AS, Thomas BL, Krishnan NC (2021) CASAS. <https://casas.wsu.edu/datasets/>. Accessed: 12 Jul 2022