# LSTM for occupancy detection in smart homes using indoor climate data

Linus Blomqvist
*dept. Information Systems and Technology (IST)*
*Mid Sweden University*
Sundsvall, Sweden
Libl1600@student.miun.se

***Abstract*—With smart homes becoming a modern standard, the need for more Internet-of-Things (IoT) devices comes with it. A smart home should imply better comfort and therefore also reduce energy consumption. A smart device should for example be able to determine how the energy management would be controlled from our daily routines. Deep learning have opened new ways to deal with the upcoming problems within the area IoT. In this paper a model has been made to deal with the problem of detecting occupancy's in 2 different datasets that are using indoor room climate data from sensors. To do this a Long Short Term Memory (LSTM) recurrent neural network model has been implemented that uses past timesteps to make predictions on new data. Implementation of the project was carried out in the framework of PyTorch in the environment Google Colaboratory. To simulate a real-world IoT device, the model was also implemented in a Raspberry pi 3. The model has been trained and then evaluated on new data. The model got highest prediction accuracy of 84.49% on the *Occupancy Detection Dataset*.**

***Keywords*—LSTM, IoT, Deep learning, Time series**

## I. Introduction

The improvement of smart homes technologies, is an increasing demand for smart devices in indoor environments. All these smart devices are collaborating with each other and are connected to the internet, that is called the Internet-of-Things (IoT) [1]. A lot of these tiny embedded devices are sensors that enable new types of intelligent applications such as smart heating, lightning and metering in smart homes. Data that is gathered from these devices, like a smart heating application, wants to represent the current state of the environment with the help of temperature data. The collected data could later be used to regulate ventilation and heating in a smart home. In the future these intelligent devices can create more sustainable homes by saving energy. One way to do this is to determine the number of occupants in an environment and later use this information to regulate an optimal temperature in the smart home.

With more IoT devices and sensors the more information they will produce. This will lead to problems in the future to store all this data. Within IoT, the research about deep learning technology has become an interesting new field to avoid storage of large data volumes. Normally deep learning uses high end graphics card and hardware. IoT devices are cheaper and use weaker hardware. It would be beneficial if this technology would be applicable for these applications.

In the early days of artificial neural intelligence, the Feedforward Neural Network was a popular Artificial Neural Network (ANN) [2]. The disadvantage of the ANN was that it was unable to capture temporal dependencies, that are dependencies that change over time. Modeling temporal data, as in a time series, allows us to predict from past data. So with applying recurrence to ANN's made it possible to capture temporal dependencies. With Time Delay Neural Networks (TDNN) it was possible for the first time to add memory to neural networks [3]. TDNNs had the advantage of looking beyond the current timestep, because its input was from past time steps. The disadvantage with this approach was that the temporal dependencies had a limitation of a short memory cell. Later came the Elman and Jordan network known as simple Recurrent Neural Networks (RNN) [4]. These networks had the issues with the vanishing gradient problem, which means that the learning for the models becomes non-existent. So it became difficult to capture relationships more than a few timesteps. In 1970 the Long Short Term Memory (LSTM) artificial RNN architecture was invented by Sepp Hochreiter and Jürgen Schmidhuber to deal with the vanishing gradient problem [5]. LSTM takes in state variables from previous LSTM nodes to remember past timesteps. These state variables can be kept fixed by gates that in the future will be reintroduced or not. As in figure 1, the LSTM gates are as follows:

- Learn gate (input gate): Combines the short-term memory and an event and keeping the most important part and ignores some bits of it.
- Forget gate: Was introduced later to the LSTM architecture. This gate decides to use or forget the long-term memory from previous cell state.
- Remember gate: Combindes the long-term memory from the forget gate with the short-term memory from the learn gate.
- Use gate (output gate): Creates a new short term memory that is the output from the long-term memory from the forget gate and short-term memory from the learn gate.

At this state the research area of LSTM networks have been

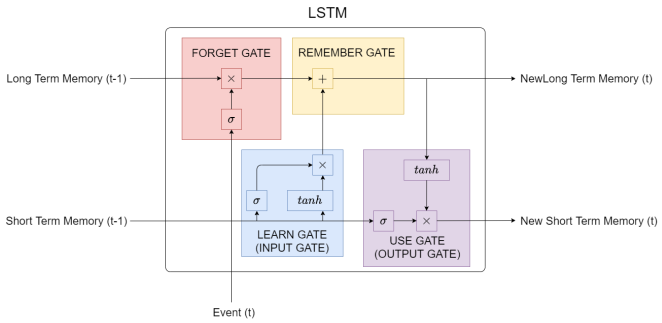


Fig. 1. LSTM architecture

within recognition, classification, prediction and sequence generation. This technology could be a potential tool for smart thermostats in a smart home to determine how many occupants there are in a room at different times or detect occupants in a room from temperature and humidity sensor data. With this information about the occupancy detection, it would decrease the energy consumption by controlling a smart home's heating or air conditioning with a suitable indoor climate temperature for the occupants. This study will investigate if it's possible to accomplish that kind of task.

## II. Purpose

The purpose of this study is to investigate the indoor climate data through a smart application. To do this a suitable LSTM model, using temperature and humidity sensor data, needs to be designed. With the model it will be possible to predict how many occupants are present in an environment at a certain time.

## III. Related work

Haider K. Hoomod and Zahraa Sabeeh Amory at Mustansiriyah University, used LSTM technique for temperature prediction for IoT room controlling application [6]. Their results showed that there model could forecast temperature data with a fixed length of values. Philipp M, Christian M, Matthias R, Björn E, Christian R, Frederik A and Zinaida B showed with a trained Naïve Bayes machine learning classifier that it was possible to predict the presence and actions (reading, working on a PC, standing and walking) of room occupants using room climate data (temperature and relative humidity) [7]. Diana Hintea, James Brusey and Elena Gaura studied several machine learning methods for estimating cabin occupant equivalent temperature from a minimalistic set of inexpensive cabin environmental sensors [8]. Philipp Morgner , Christian Muller , Matthias Ring , Bjorn Eskofier , Christian Riess , Frederik Armknecht and Zinaida Benenson develop a method for predicting the optimal indoor air temperature [9]. Yafei Zhao, Paolo Vincenzo Genovese and Zhixing LiThis proposed a building intelligent thermal comfort control system based on the Internet of Things and intelligent artificial intelligence [10].

## IV. Method

The datasets that will be used is *Occupancy Detection Dataset*[1] [11] and *Room Climate Datasets*[2] [7]. *Occupancy Detection Dataset* contains collected temperature, relative humidity, light and CO2 sensor data. The experimental data was used for a binary occupancy classification to see if there were occupants in a room or not. The occupancy data was collected from timestamped pictures that had been taken every minute. *Room Climate Datasets* contains almost 90 hours of collected room climate data at three different locations in form of controlled experiments. Each location was equipped with 3 to 5 room climate sensor nodes at different positions. Each node has the ability to collect temperature (°C), relative humidity (%) and two different light (wavelength, nm) sensors. In these experiments 0, 1 or 2 occupants have been in the room at different times during almost 40 hours of the dataset. This study will use one node from each of the 3 different rooms in the *Room climate dataset*, to deal with the multivariate time series problem because otherwise it will be too much noise with all the sensors combined from each room as figure 2 shows. Since *Occupancy Detection Dataset* only consists of one sensor value for the temperature and humidity data, no filtering sensors had to be made. The datasets were normalized with standardization. This will scale each feature to have zero mean and a standard deviation of one.

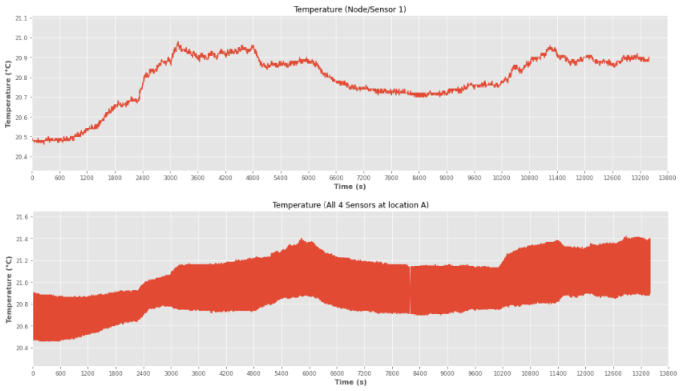


Fig. 2. Sensor A1's plot versus the plot of all 4 sensors at location A

The model will be trained and tested for each of the locations to evaluate if the accuracy differs at the different location. To train the model on the two datasets, each dataset will be split up into 2 datasets. The first dataset will be the training dataset and second dataset will be the validation dataset. The validation dataset will be used under the training phase to see that the model does not under- or overfits on the training dataset. At last the model will be evaluated a separate test dataset. The test dataset will simulate a real-world environment that the model has not seen before.

[1]Occupancy detection dataset repository
[2]Room climate dataset repository

### A. Parameters

In this study, the model will be implemented with Python's deep learning framework and scientific computing package PyTorch. The design of the LSTM recurrent neural network will have 3 layers, 1 input layer, 1 hidden layer and 1 output layer as shown in figure 3 below.

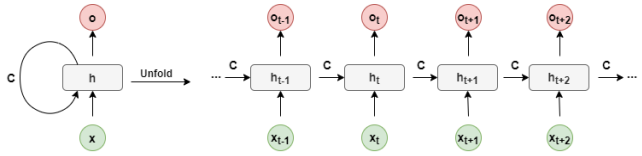


Fig. 3. Where $x$ (green color) is the input layer, $h$ (grey color) is the hidden layer, $o$ (red color) is the output layer, $C$ is the communication between each block and $t$ is the timesteps.

The input layer of the model will have the input shape of [batch size, timestep, input size], there the *batch size* is a mini batch of $n$ samples, *timestep* will be a experimental value between 10-400 because LSTM models won't perform that well with higher timesteps (between 200 to 400) and the *input size* is the 2 features, temperature and humidity. The *hidden layer* is a hyperparameter that also needs to be experimented with different values to control that the learning process will go in the right direction. The *output layer* will make that the classification will be either 0, 1 or 2 for the *Room climate dataset* or 0 or 1 in the *Occupancy Detection Dataset* depending on how many occupants there are in the room at a specific time. The model optimizer is the Adam optimization algorithm, because it learns the *learning rate* by itself, which will lead to that the scheduler won't be needed as in the example Stochastic gradient descent (SGD) [12]. To classify the 3 different outputs (occupancy 0, 1 and 2) or the binary problem, the model will use a Cross Entropy Loss for both datasets. The PyTorch implementation of Cross Entropy loss applies both Softmax and Negative log likelihood which means that the classification occupancy does not need to be one-hot encoded. Cross Entropy loss is used for multiclass classification problems but can also be used with binary classification problems. In this paper it will be used when training on the both datasets. The parameters for the model is shown in table 1 below for the two datasets.

TABLE I
LSTM MODEL PARAMTERS FOR THE TWO DATASETS

| Parameter | Room Climate | Occupancy Detection |
|---|---|---|
| Classes | 3 | 2 |
| Mini batch size | 32 | 32 |
| Learning rate | 0.01 | 0.01 |
| Timestep | 20 | 20 |
| Input layer size | 2 | 2 |
| Hidden layer size | 40 | 40 |
| Output layer size | 1 | 1 |
| Epoch | 15 | 15 |

The LSTM model will be implemented in Google Colaboratory or Colab in short. It is a Python development environment that runs in the browser using Google Cloud. Colab allows free access to computing resources including GPUs as for example Tesla V100-SXM2-16GB. Usually it acquires more computational power under the training phase rather than under testing. To simulate an IoT device, the model will be evaluated on a Raspberry pi 3. The reason to test the model on this device, is that the controlling unit in a smart home would be a smart device as for example a thermostat. So if the system went offline it would be beneficial if the smart device could still use the model.

To show how the prediction went for the model on the test dataset, the accuracy will be calculated with the formula:

$$Accuracy = \frac{The\ total\ correctly\ classified\ labels}{Total\ dataset}$$

## V. RESULT

The model was trained with the same epochs and mini batch size as shown in the table 1. In figure 4 the loss functions of the *Occupancy Detection Dataset* training dataset is compared with the validation dataset under the training phase. As the loss functions converged relatively smoothly, it does not show any sign of underfitting or overfitting.

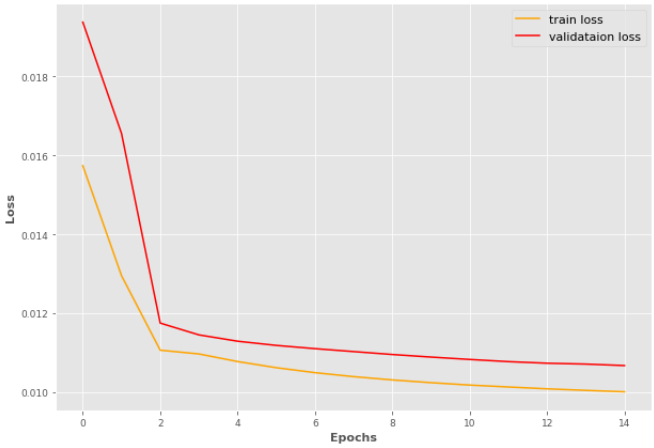


Fig. 4. Loss function for the training and validation dataset

### A. Model evaluation

After the training phase, the model that trained on the *Occupancy Detection Dataset* got the highest validation accuracy of 86.41%. Then the model got evaluated on the test dataset for each of the two datasets. The model with the best validation accuracy got the highest test accuracy of 84.49% on the *Occupancy Detection* test dataset. The low accuracy on the *Room climate dataset* could have been caused by the lack of exploration of the dataset due to time constraints. The results that can be drawn from the table 2, is that the model was able to have a high accuracy when dealing with the binary occupancy detection problem.

Evaluating the model that got the highest accuracy on the test dataset from the *Occupancy Detection Dataset*, a comparison of the predictions against the true labels form the dataset can be viewed in figure 5. With the hyperparamters that

TABLE II
ACCURACY IN PERCENT FOR SENSOR A1, B1 AND C1 FROM *Room climate dataset* AND THE SENSOR FROM *Occupancy Detection Dataset*

| Sensor | Validation accuracy | Test accuracy |
|---|---|---|
| Room climate dataset A1 | 61.75 | 57.32 |
| Room climate dataset B1 | 59.87 | 56.72 |
| Room climate dataset C1 | 63.31 | 58.43 |
| Occupancy Detection Dataset | 86.41 | 84.49 |

were used in the paper the model made a good approximation of the test dataset from the relative small training dataset. The test dataset from the *Occupancy Detection Dataset* were used to test the model on a Raspberry pi 3. The evaluation of the model took approximately 1 min. The Raspberry pi 3 had no problem under the process but when testing with higher timesteps it struggled with overheating and needed cooling. This indicates that the model could be used in a real-world IoT smart application.

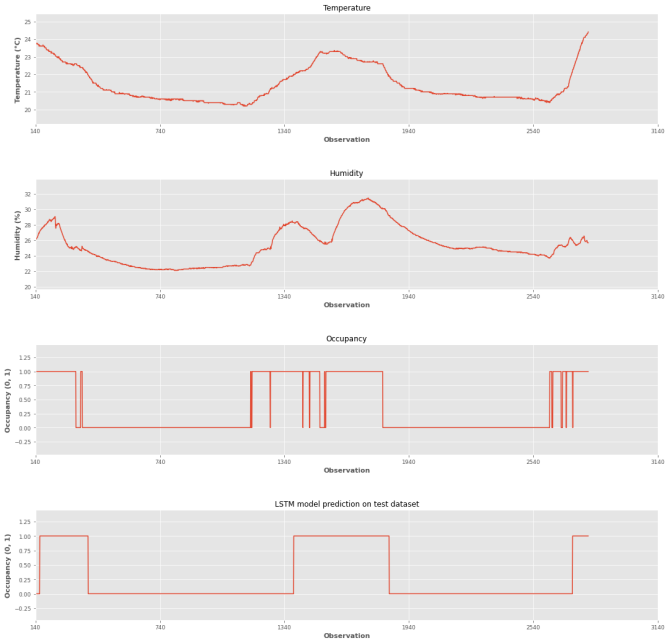


Fig. 5. Sensor data from *Occupancy Detection Dataset* and the predictions from the LSTM model

## VI. CONCLUSION

In this paper a LSTM recurrent neural network model has been implemented in Pytorch to deal with the time series sequence to label prediction problem. The problem was to predict the occupancies in a room with collected room climate data from temperature and humidity sensors. The model used 20 time steps that was later used to remember older states when it were tested on a test dataset from *Occupancy Detection Dataset*. The model had 84.49% accuracy on the test dataset, that resulted in a good approximation when detecting occupants in a room. The model was also implemented on a Raspberry pi 3 and were able to run with no problems. This shows that the model could be a potential tool for IoT devices as for example a smart thermostat in a smart home to detect occupants in a room at different times using temperature and humidity sensor data. The ability to detect occupants will lead to better regulation of indoor climate and there by better comfort and energy reduction.

### A. Ethical and societal issues

From a ethical point of view, this approach touch on the issue of privacy. Could this data be used by large companies that use this technology in their product. Another issue of privacy is if the temperature and humidity sensor data could be used by an attacker to determine how many occupants there are in a room and use it for personal gain.

### B. Future work

For future work, it would be beneficial to analyze more or use a number of statistical descriptors on the *Room climate dataset* to be able to get better results when training the model. Due to time constraints, of the project, the cleaner *Occupancy Detection Dataset* had to be added to show that the model was able to handle the occupancy detection problem. A larger dataset or more climate data scenarios of different indoor environments would strengthen the model and improve the accuracy. More experimental with the hyperparameters could also improve the model prediction accuracy.

## REFERENCES


[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] Murat Hüsnü Sazlı. A brief review of feed-forward neural networks. 2006.

[3] Masahide Sugiyama, Hidehumi Sawai, and Alexander H Waibel. Review of tdnn (time delay neural network) architectures for speech recognition. In *1991., IEEE International Sympoisum on Circuits and Systems*, pages 582–585. IEEE, 1991.

[4] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Haider K Hoomod and Zahraa Sabeeh Amory. Temperature prediction using recurrent neural network for internet of things room controlling application. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 973–978. IEEE, 2020.

[7] Philipp Morgner, Christian Müller, Matthias Ring, Björn Eskofier, Christian Riess, Frederik Armknecht, and Zinaida Benenson. Privacy implications of room climate data. In *European Symposium on Research in Computer Security*, pages 324–343. Springer, 2017.

[8] Diana Hintea, James Brusey, and Elena Gaura. A study on several machine learning methods for estimating cabin occupant equivalent temperature. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 629–634. IEEE, 2015.

[9] Yafei Zhao, Paolo Vincenzo Genovese, and Zhixing Li. Intelligent thermal comfort controlling system for buildings based on iot and ai. *Future Internet*, 12(2):30, 2020.

[10] Jing Jin, Shaolong Shu, and Feng Lin. Prediction of indoor air temperature based on deep learning. *Sensors and Materials*, 31(6):2029–2042, 2019.

[11] Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.