# Report Simula Summer Internship

This report describes the process of applying a number of Machine Learning techniques to a Smart Home dataset [link to the dataset]. The dataset [1] consists of labeled data gathered in two houses with a number of binary sensors attached to various appliances and objects around the house. First, LOF and isolation forests were used to detect anomalies in the sensor values by combining features and sensor values. These methods worked well for sensors which did not have a very large variance in the features used for similarity measures.

Next, RNNs - more specifically stacked Long Short-Term Memory networks - were used to predict future sensor values. Single step and multi-step predictions were attempted, with single-step prediction showing promising performance with accuracy > 90% and few false positives for sensors with a low number of activations. The models could be improved given more training data.

Finally, the best model from the prediction section was used to synthesize 10 days of new sensor data. The model uncertainty was evaluated by performing a few hundred simulation runs while adding gaussian noise to the model output. Visually, the synthesized data is similar to the training data. However, entropy and autocorrelation plots show that there is still room for improvement.

[1] Activity Recognition Using Semi-Markov Models on Real World Smart Home Datasets - T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse

## Data exploration

The dataset consists of 2 separate measurements, done in different houses with different subjects.

Each house has sensors installed in 2 rooms (bathroom and kitchen). Examples of sensors are Microwave sensor, Toilet flush sensors, doors open/close sensors.

The dataset contains start times and end times of each sensor activation.

The dataset also contains activity labels which the subject has recorded as he was performing daily tasks (brushing teeth, cooking, etc.)

A typical day looks something like this: some activity when the subject wakes up (0900-1000), nothing until he returns from work (1700) and more activity when he goes to sleep (2300)

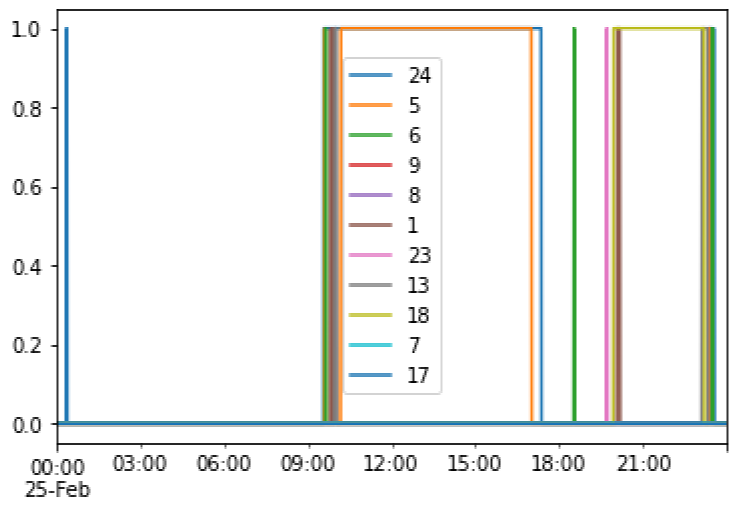
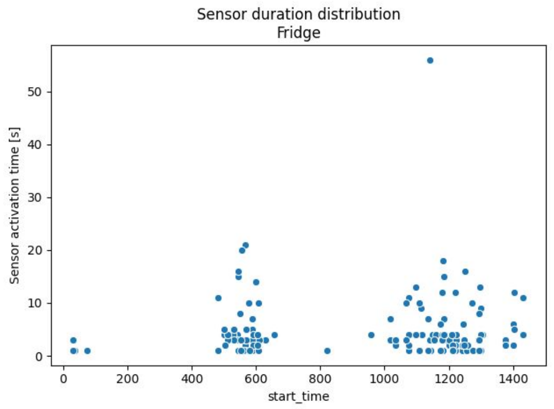


*Figure 1 - All sensor activations on 25-Feb in House 1 dataset.*

Some sensors (24, 5) are activated for a longer time. These are door sensors - apparently doors are often left open as he goes to work. Most other sensors have shorter activation times, such as the fridge sensor below.

**Fridge sensor activation times and sensor values over time**

*start_time=minutes past midnight*



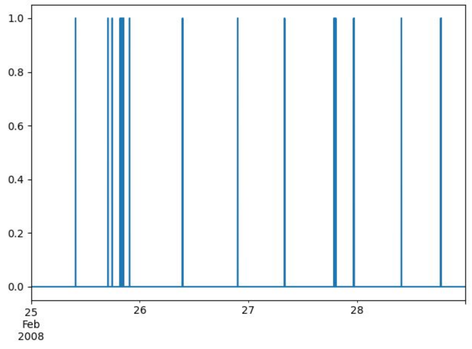


*Figure 2 - Scatter plot of fridge sensor duration and start time.*

*Figure 3 - Fridge sensor activations*

## Anomalies

A possible application of machine learning to this dataset is to identify and possibly even predict anomalies.

There are a number of possible anomaly types

1. Sensor duration is longer/shorter than normal (see fridge with duration=55)
2. Sensor is activated at a time which has not been seen before (fridge with start_time = 800 minutes past midnight)
3. Sensor is activated with unusual frequency - not activated at all, or activated much more frequently than normal

Unfortunately, this dataset does not have any labeled anomalies. This does not mean that there aren't any - the fridge sensor readings shown above contain some readings which are very different from other readings for that sensor.

### Anomaly prediction and detection

#### LOF with similarity measures

Similarity measures between sensor values can be used to evaluate how similar a sensor reading is to sensor readings completed in the past. If this particular sensor reading has never been seen before, it might be anomalous or abnormal. Once a method to measure similarity between sensors has been established, we can use a technique such as LOF (Local Outlier Factor) to detect outliers. LOF works by comparing the local density of a point to the local density of the **k** nearest neighbours, where **k** is a hyperparameter. In the fridge sensor example above, the sensor reading with start_time 1150 and activation time 55 has a very low density compared to its nearest neighbours and will probably be marked as anomalous.

Many sensors seem to have a pattern which they repeat daily, such as the fridge sensor above (activation at 600 and 1200) minutes past midnight.

Multiple similarity measures were tried

- Only duration
- Only start_time
- Combination of duration and start_time

### Similarity measured using sensor duration

The results are shown visually below.

**Anomaly detection using only sensor duration**

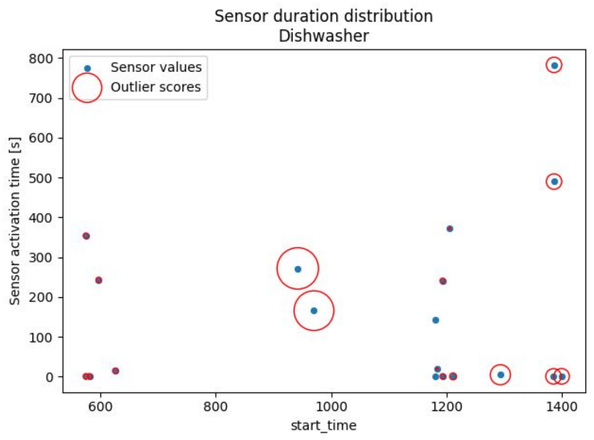


*Figure 4 - Dishwasher sensors with outlier scores*

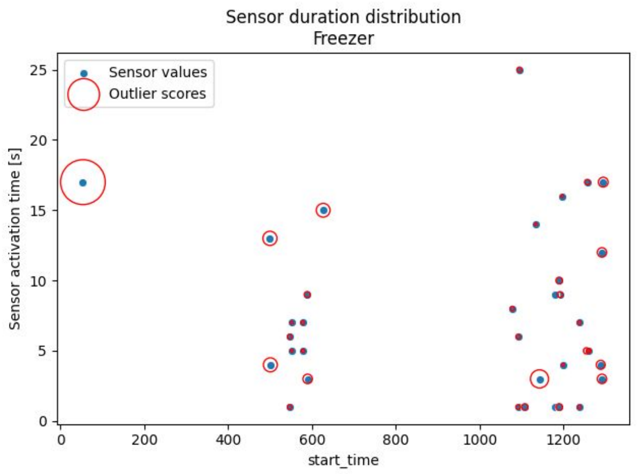


*Figure 5 - Freezer sensor with outlier scores*

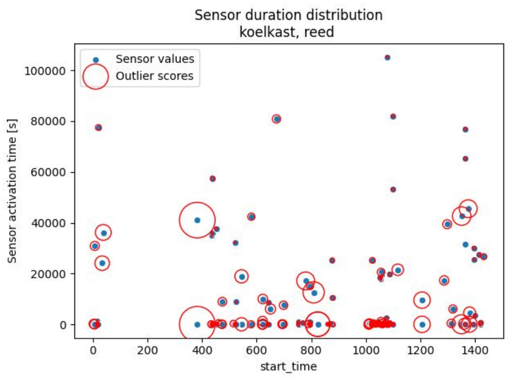


*Figure 6 - fridge sensor in House 2 with outlier scores*

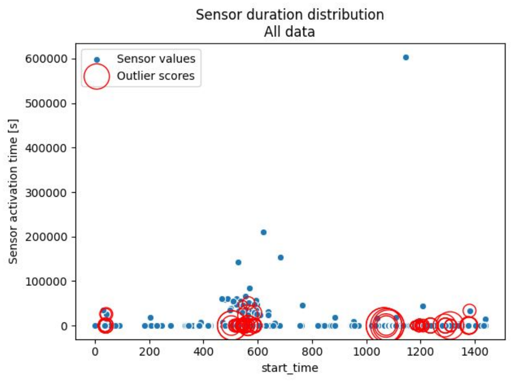


*Figure 7 - all sensors activations in House 1*

The third figure (koelkast, reed) has a mix of very long activation times and very short activation time. This gives less clear results when using duration as a similarity measure. This sensor is probably faulty, as this is a fridge sensor which should probably not be open for more than a few seconds at a time.

## Similarity measured using sensor duration and start time

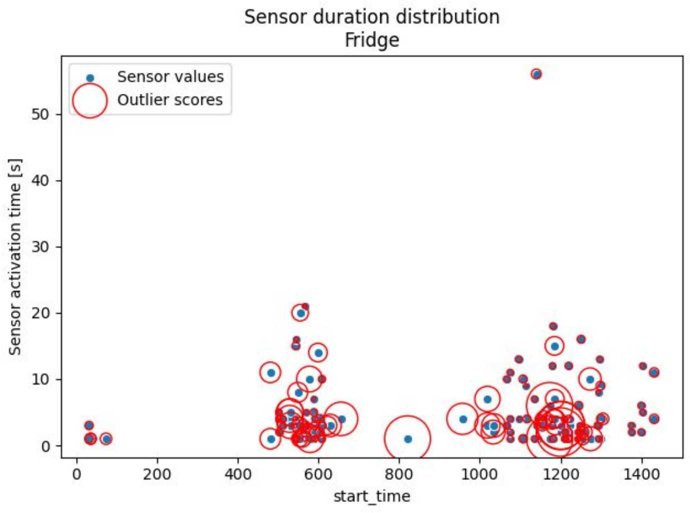


*Figure 8 - fridge sensor in House 1 with with outlier scores, calculated using duration and start time features*

Calculating similarity with a combination of duration + start time does not perform well for all sensors. I scaled the duration using a min-max scaler and calculated the euclidean distance between different points. For sensors where the durations fall into a narrow range, such as the Dishwasher sensor, some values which seem to be anomalous are identified as outliers. For other sensors, such as the fridge sensor above the combining the two features and using LOF does not work very well to identify outliers. Values which are clearly dissimilar to other values are marked as inliers.

Modifying the number of nearest neighbours used for LOF has a big impact. Left = 5 neighbours, right = 10. I think there are not enough data points to use LOF with this many neighbours.

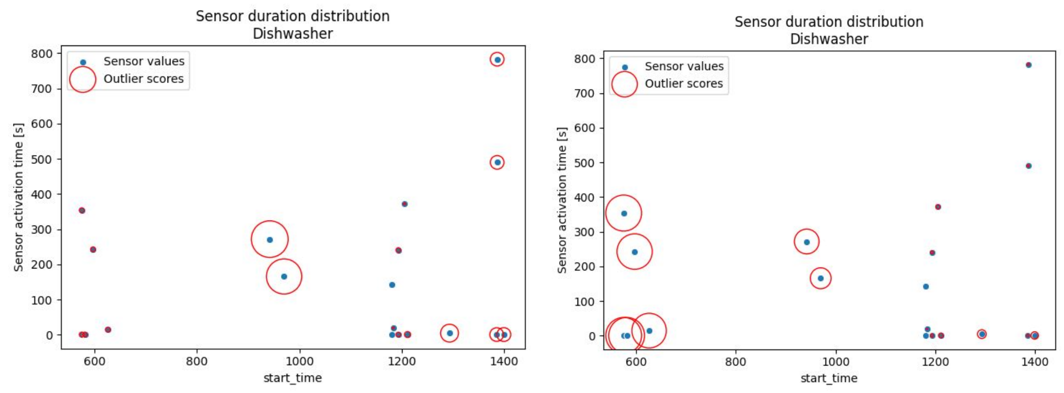


*Figure 9&10 - dishwasher sensor in house with with outlier scores, calculated using LOF with k=5 and k=10 respectivelty*

Anomaly prediction could be refined by using other features, such as

- Time since previous activation
- Similarity between daily activation patterns, maybe using dynamic time warping. This will only be possible for frequently activated sensors - others may not have enough data. See the table below which shows that the dishwasher sensor was only activated 19 times over the course of 28 days.

## Isolation forests

Instead of using LOF, I could also try to detect anomalies with isolation forests.

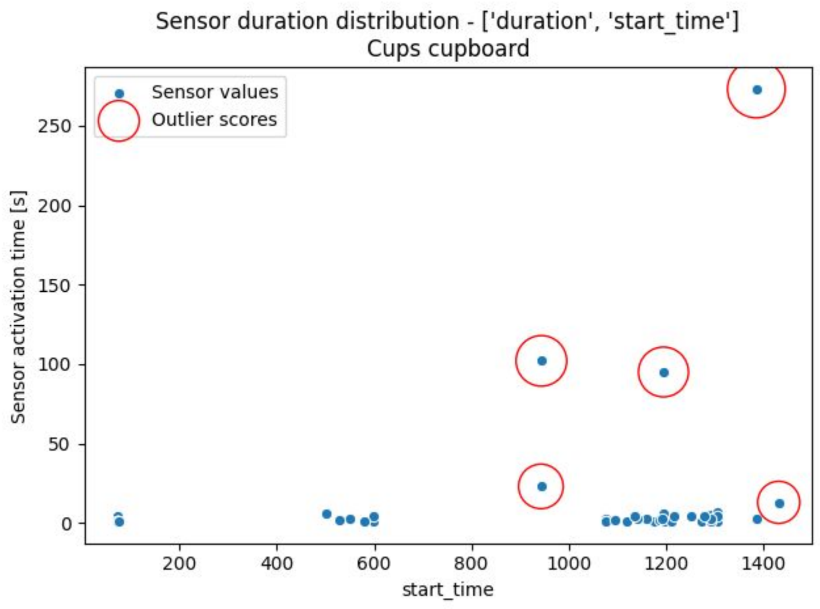


*Figure 11 - outlier scores calculated using isolation forests with duration and start time features for cups cupboard sensor.*

After trying different threshold values for the decision function, this method seems to result in less false positives. I would advise using isolation forests with a manually tuned decision threshold over LOF, especially for sensors with fewer activations.

### HMM

Another option is to use Hidden Markov Models to learn transitions between sensor states. I can discretize the dataset into 60 windows and try two different types of encodings of the dataset: last-fired (from the original paper) or simply raw sensor values.

### LSTM

LSTMs can also be used to learn patterns in time-series. This can then be used to predict anomalies by calculating the residual sensor values. If they exceed some threshold, something is probably wrong.

I did not focus on anomaly prediction with LSTMs and instead moved on to generating synthetic time series with RNNs.

# Machine Learning in prediction

In addition to anomaly detection, another way we could apply Machine Learning techniques to the dataset is by predicting sensor values. If we can accurately predict future sensor values, this could also be used to detect anomalies - if the actual sensor value is not the same as the prediction, something may be wrong.

There are different methods to predict future values of a time series given a set of past values. One option would be to use probabilistic models, such as ARIMA or a Hidden Markov Model. Neural networks can also be used to predict future values of a time series. Two types of neural networks that can be used to do this are Recurrent Neural Networks and Convolutional Neural Networks. I decided to use an LSTM network as it is able to predict arbitrary length sequences with an arbitrary number of input features. Unlike ARIMA and HMMs, the network can learn non-linear relationships between sensors.

## Methodology

The data for each sensor was first encoded into a continuous time series measured at a fixed interval. If the interval falls within a sensor’s activation period (start time -> end time) then the value for that interval is set to 1, else 0.

Different models were built in Keras and used to predict future sensor values. The data was first reshaped into the required shape (LSTMs and GRUs require the data to be shaped into [#Samples, #Past time steps, #Features]).

Evaluation of the models was done by splitting the data into a train, validation and test set. The test set consisted of the last 2 days of data, the validation set was the 2 days before that.

## Model Design

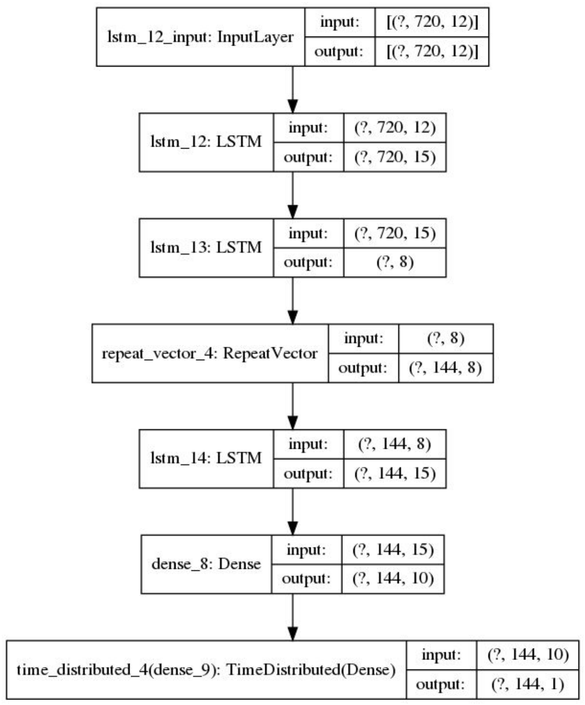
A number of different model designs which used LSTMs were evaluated. Most of the designs were variations of the following two types:

- Encoder-Decoder
- Vector Output

### Encoder-Decoder

This type of network consists of an encoder and a decoder. The encoder is a stack of LSTMs and a hidden layer which outputs a vector that encapsulates the input information into a single vector. The decoder receives this encoding of the input and uses a stack of LSTMs to make a prediction of the output value at different time steps. The input and output vectors do not have to be the same length or shape.

***Encoder decoder model for synthetic time series generation***.
*LSTM_12 and LSTM_13 are the encoding layers, while LSTM_14, Dense_8 and the time distributed Dense_9 are the decoding layers.*



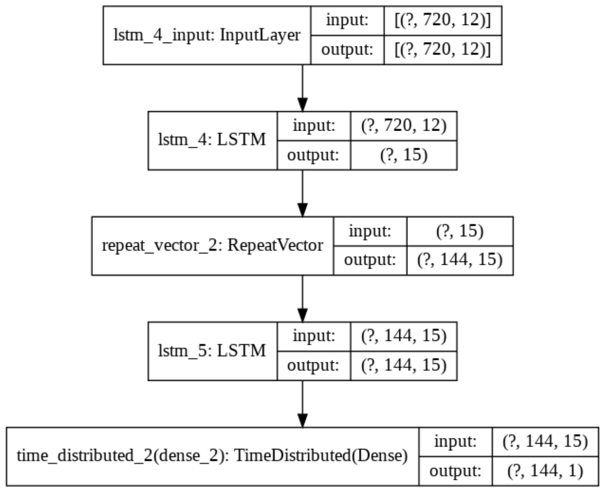


*Figure 12 & 13 - Encoder decoder network designs.*
*The right model is a simpler version of the left model, with the same encoder/decoder structure*

## Vector output

A vector output model does not have an encoding/decoding structure. Instead, the output of a number of stacked hidden LSTM layers can be directly interpreted as a multi-step prediction.

**Vector output for synthetic time series generation**

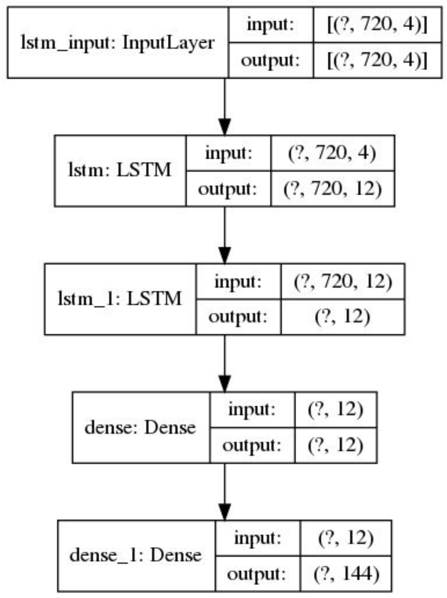


*Figure 14 - Vector output model design*

## Model evaluation

I chose to limit the sensors used for training to sensors 24, 6, 5 and 9 from the first dataset. These are sensors which range in the relative number of activations. The dataset from the second was not used as training the networks took a very long time on my machine.

**Table containing performance metrics after training and evaluation on test set**

| **Model name** | **Model #** | **Sensor id** | **TN** | **FP** | **FN** | **TP** | **Mean binary accuracy** |
|---|---|---|---|---|---|---|---|
| encoder-decoder | 5 | 5 | 18.31 | 38.50 | 16.06 | 47.13 | 0.55 |
| lstm_vector_output | 7 | 5 | 30.69 | 26.13 | 11.48 | 51.71 | 0.69 |
| Single step vector output | 33 | 5 | 92 | 7 | 3 | 90 | 0.95 |
| encoder-decoder | 4 | 6 | 73.54 | 6.23 | 35.75 | 4.48 | 0.65 |
| GAN | 10 | 6 | 79 | 0.75 | 39.8 | 0.25 | 0.66 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| lstm_vector_output | 6 | 6 | 69.90 | 9.88 | 35.69 | 4.54 | 0.62 |
| Single step vector output | 30 | 6 | 173 | 7 | 11 | 1 | 0.90 |
| encoder-decoder | 8 | 9 | 109.96 | 2.65 | 7.17 | 0.23 | 0.92 |
| lstm_vector_output | 9 | 9 | 108.15 | 4.46 | 6.90 | 0.50 | 0.91 |
| Single step vector output | 31 | 9 | 184 | 7 | 0 | 1 | 0.97 |
| lstm_vector_output | 0 | 24 | 26.04 | 7.85 | 33.77 | 52.33 | 0.65 |
| encoder-decoder | 3 | 24 | 24.17 | 9.73 | 9.50 | 76.60 | 0.84 |
| Single step vector output | 32 | 24 | 87 | 4 | 9 | 92 | 0.93 |

### Multi-step Time-series Prediction

The data is discretized into 600s bins. It turns out that the numerical instability I was encountering was due to the use of the ReLU activation function in the hidden layers. By changing this to the hyperbolic tangent activation function, the numerical instability seemed to disappear.

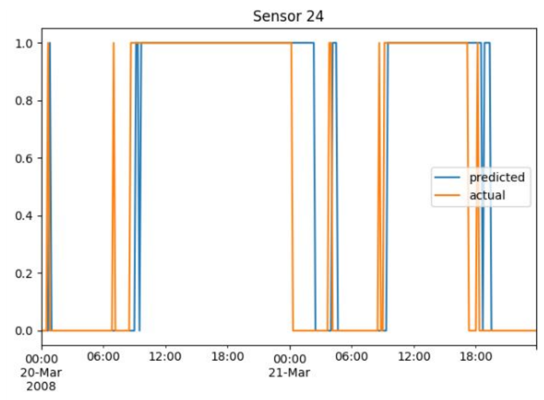
Smaller bin sizes lead to a decrease in model performance, as more timesteps need to be predicted at once.

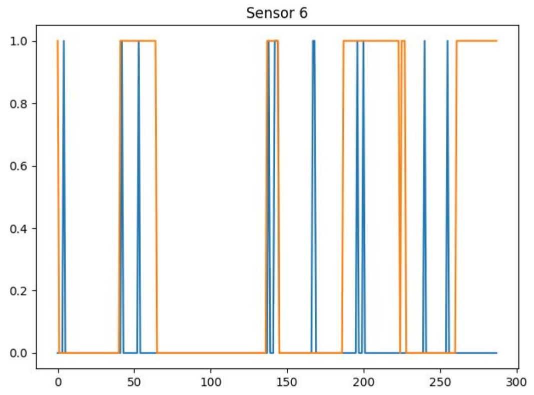
Training takes approximately 15s per epoch, and a separate model is trained for every sensor. All sensor inputs and the time in hours are used as input features (11 sensors + 1 time in hours).

Performance is increased by including hour and training a model for every sensor separately using class weights for sensors with very few activations. See table in document Model Results for more information about performance.

Some sensors have very few activations. This makes it more difficult to predict exact activation times as there is not much data (only 30 activations in total). Other sensors, such as sensors 24, 5, 6 have longer activation times which makes prediction and generating synthetic time series easier.

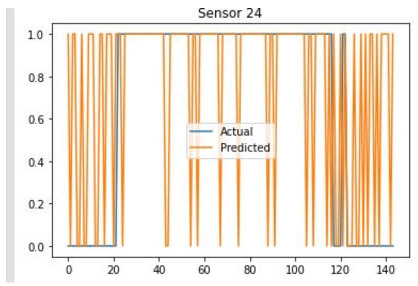
**Examples of predicted time series for sensors 24 and 6**

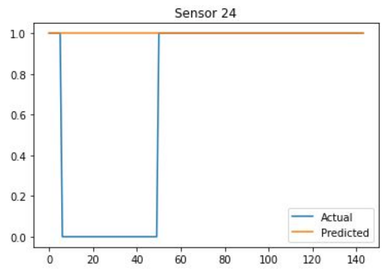


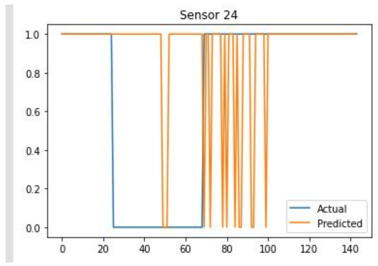


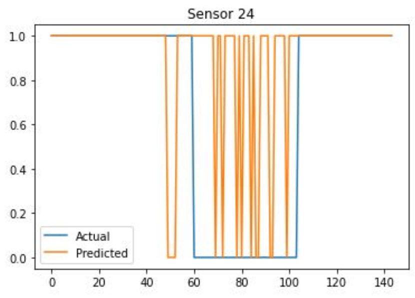
*Figures 15&16 - Predicted activations for sensor 24 and 6..*

Some patterns are predicted correctly such as below but this is mostly luck - the exact same prediction is repeated for many input samples. (see for example predictions 3 and 4 below - they have different input samples but the same prediction)









*Figures 17-20 - Predicted activations for sensor 24.*

As per observation this is mostly due to a lack of training data - there were only 3000 training samples which is generally not enough to learn a complex model.

The performance for sensor 9 which has fewer activations is quite poor - the mean number of false negatives was between 6-8 while the number of true positives was between 0.2-0.5. This means that in most cases, the sensor activation was missed while many incorrect activations were predicted.

This could be caused by the loss function that was used - binary cross entropy. Sensor 9 has an average of around 1 activation per day, which means that there is a large imbalance in the dataset. The loss function without sample weights does not account for this.

### Predicting single time step ahead

In the predictions above, I attempted to predict time series by generating the entire new time series at once. However, another possibility is to predict one time step ahead, and use this new predicted value to predict another time step ahead, etc.

To compensate for the imbalance in the dataset (many more sensor values are 0 than 1 for sensors 6, 9) a weighted version of binary cross-entropy was used. Sample weights in batches were set to the #sensor values=0/#sensor values=1. This prevents the model from always predicting sensor value = 0, which would lead to a high accuracy but a low performance.

This approach has a higher accuracy than predicting many steps ahead (see results in Model Results document). Even though the time step size was lower than before (15 minutes compared to 1 hour) a substantial improvement is visible across all metrics (false positives, true positives, false negatives, true negatives and accuracy). Even for sensor 9 which has fewer activations there is an improvement in accuracy, as the model manages to correctly predict the activation time of the sensor in the test set.

A model which only needs to predict a single time step ahead does not need to be as complex as a model which predicts many time steps ahead. Due to a lack of training data (2000-3000 training samples) the model needed to be kept as simple as possible to prevent overfitting. Perhaps with more training data, the models which can generate many future time steps at once would perform better.

## Discourse factors

The model which predicts a single time step ahead performs substantially better compared to the model which predicts multiple time steps at once. As per observation, this may be related to the scarcity of training data. This is a common problem when dealing with cyber physical systems, and this exercise was good practice in how to deal with these problems.  The training loss figures in the Model Results document clearly show that overfitting is occurring on the training set, as the loss in the validation set continues to increase as training progresses. Further improvement of the multi-step model fell outside the scope of the exercise, which is why I focused on the single-step prediction model for the next part of the assignment.

## Data synthesis

Once a model which is able to predict future time steps of a sensor value has been trained it can be used to synthesize new data. This section of the report will describe how new time series was synthesized and how the predictive uncertainty was evaluated.

### Predictive uncertainty estimation

The model used to predict a single time step ahead from the previous section was used to perform data synthesis. This model was much more accurate than the other models that were evaluated. Since this model was only able to predict a single timestep ahead, the predicted sensor value was used as input to predict the next value. This results in a model which is able to produce arbitrary length sequences given a window of past values.

The following model on the right was used to predict 1 time step ahead given the past 96 time steps (1 day of data).

The length of the time step was 15 minutes. Training with shorter time steps or more past time steps unfortunately took too long on my machine, but may result in improved performance.

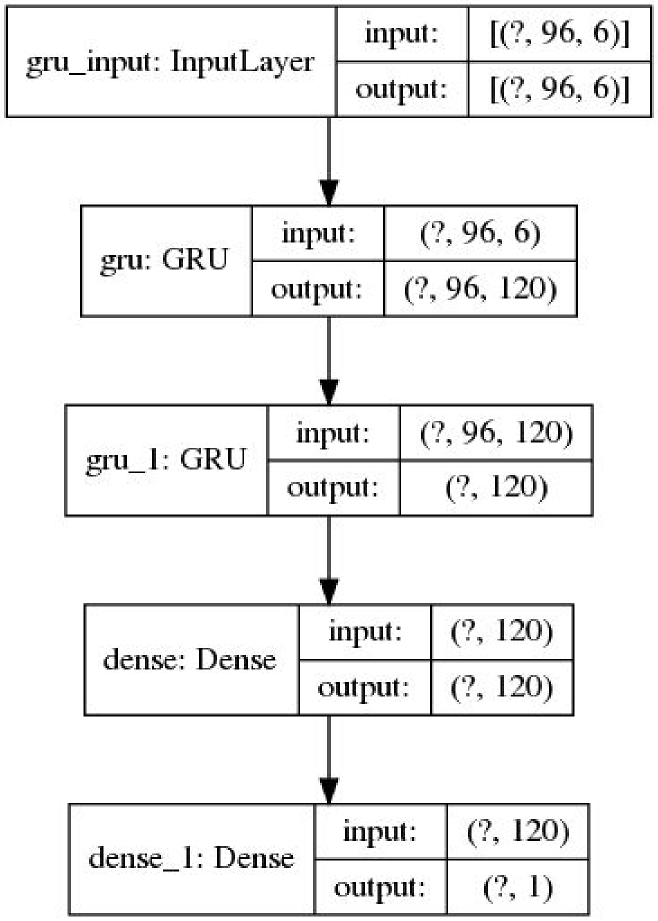


*Figure 21 - Model used for data synthesis*

The model was trained for 20 epochs. A separate model was created to predict output values for sensors 24, 5, 6, and 9. These are sensors with either many activations (sensors 24 and 5) or fewer, sporadic activations (6 and 9).

The input vector had 6 features - values for 24, 5, 6 and 9 + 2 features for time encoded into a sinusoid (sin(seconds past midnight), cos(seconds past midnight)).

The sensor values in the training set show that sensors 6 and 9 have a more sporadic, activation pattern than sensors 24 and 5. The figure that shows the mean activation pattern of the sensors in the training set clearly show that sensors 24 and 5 follow a somewhat fixed pattern, while the other sensors do not. This sporadic activation pattern is clearly visible in the autocorrelation plots here.

This means that sensors 6 and 9 will probably be more difficult to predict than sensors 24 and 5.

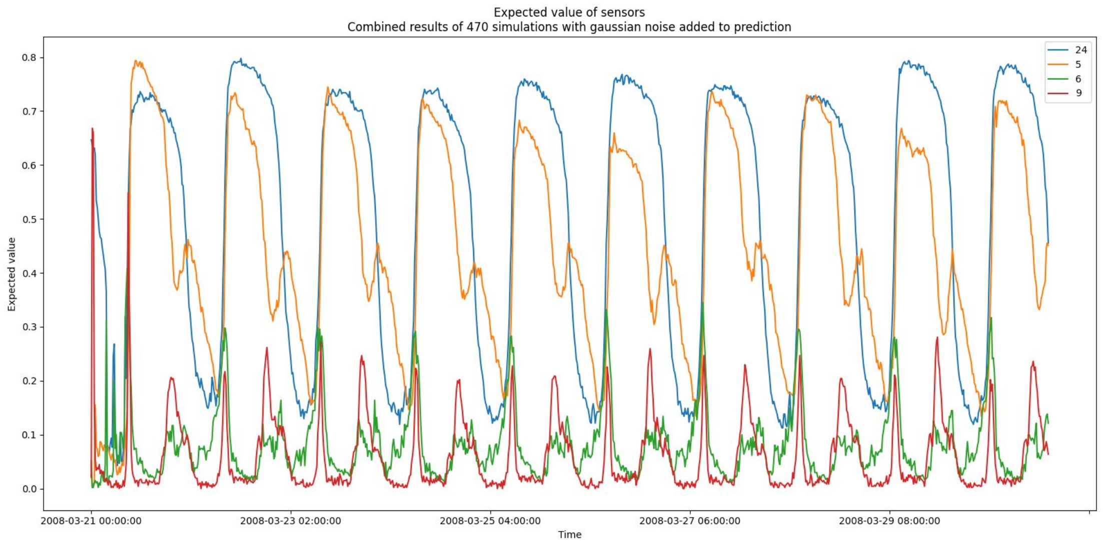
10 days of synthetic data was first generated deterministically (predicted value > 0.5 cast to 1, else 0). The data started at the beginning of the test set (which is 2 days of data) and continues 8 days after the test set has ended.

To evaluate the uncertainty, gaussian noise was added to the output prediction, and multiple datasets were synthesized. In total, 470 simulation runs were completed. The expected value of the sensors can be found in the figure below. In the second figure, the deterministic prediction and non-deterministic expectation (sum of all 470 runs for each time step/470) are also plotted.

The synthesized data will be evaluated qualitatively (visually comparing the time series to the training data) and quantitatively using autocorrelation plots and sample entropy. Autocorrelation plots will show if patterns in the training data are also reflected in the synthesized data. Comparing sample entropy for synthesized and training data will allow us to compare the complexity of the series - a sensor with many activations will have a lower complexity than a sensor which is not activated for the majority of the time.

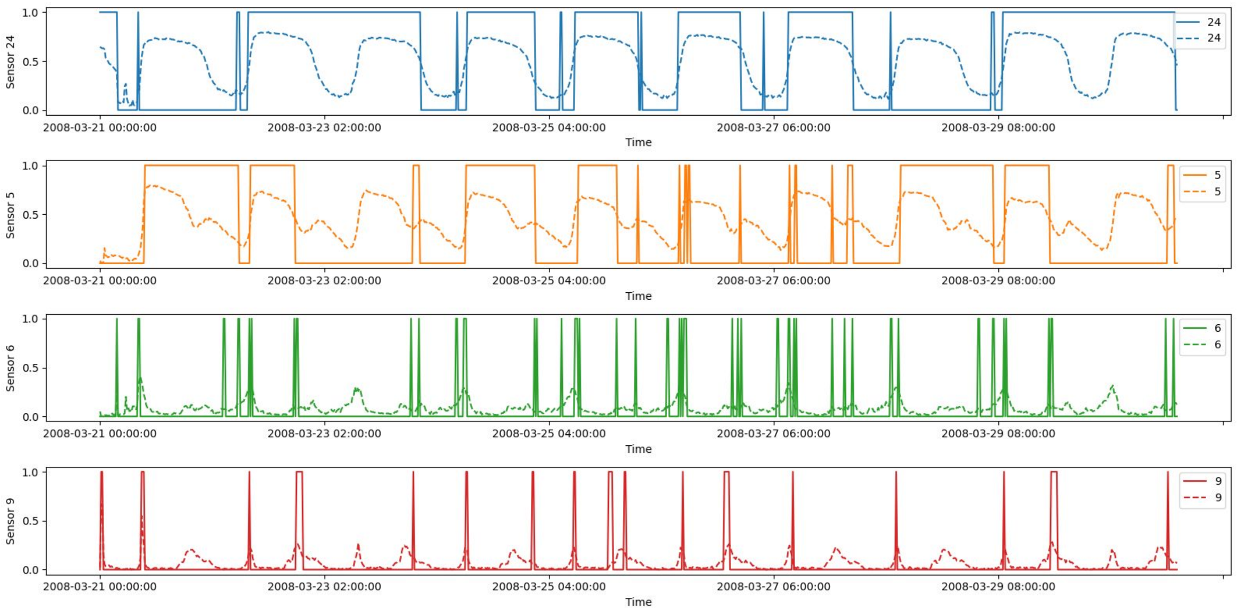
## Results

**Figure 22 - Expected value of sensors in synthetic dataset (averaged over all runs)**



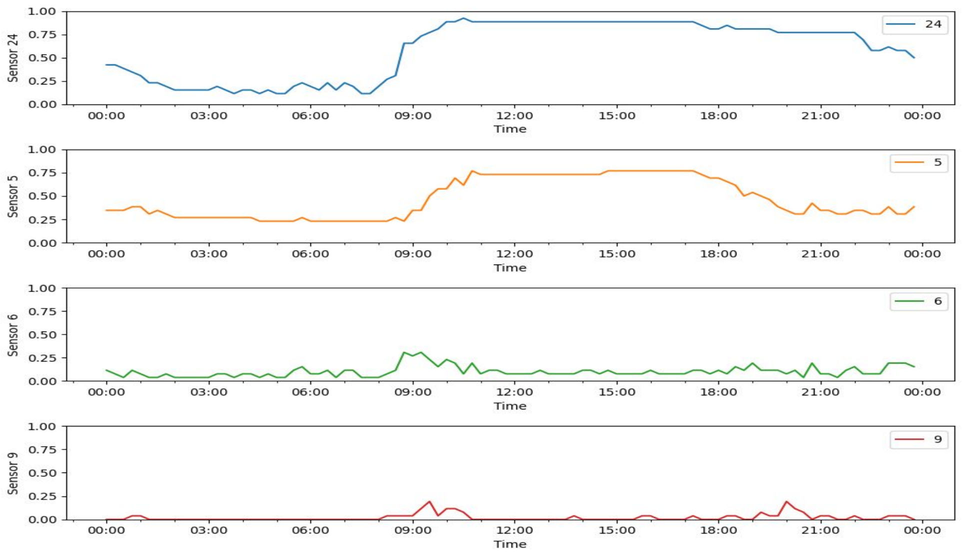
## Figure 23 - Synthesized data, 10 days

*Dashed line=uncertain prediction, solid line = deterministic prediction.*
*Gaussian noise $\mathcal{N}$(0, 0.2) added to prediction*



## Figure 24 - Mean activation pattern in training dataset

*Sensor values at Time=t averaged over 26 days of data*



## Figure 25 - All sequences in training set

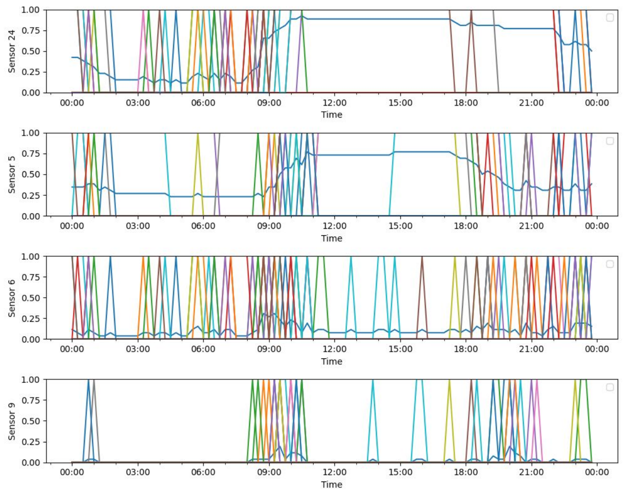
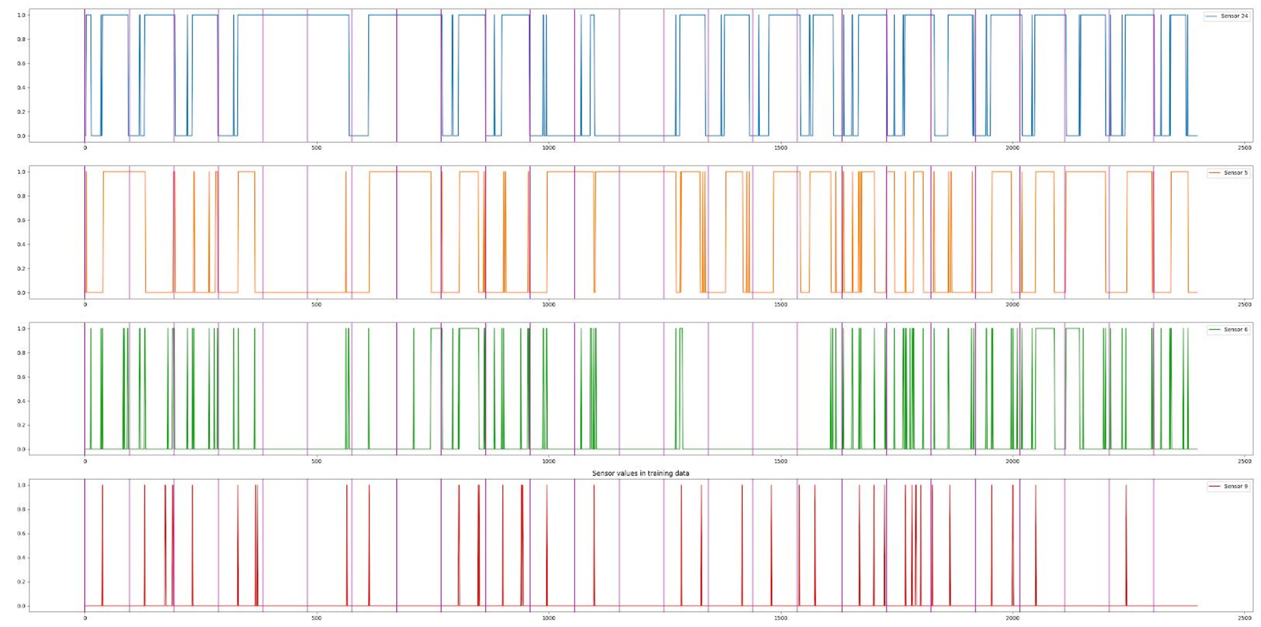
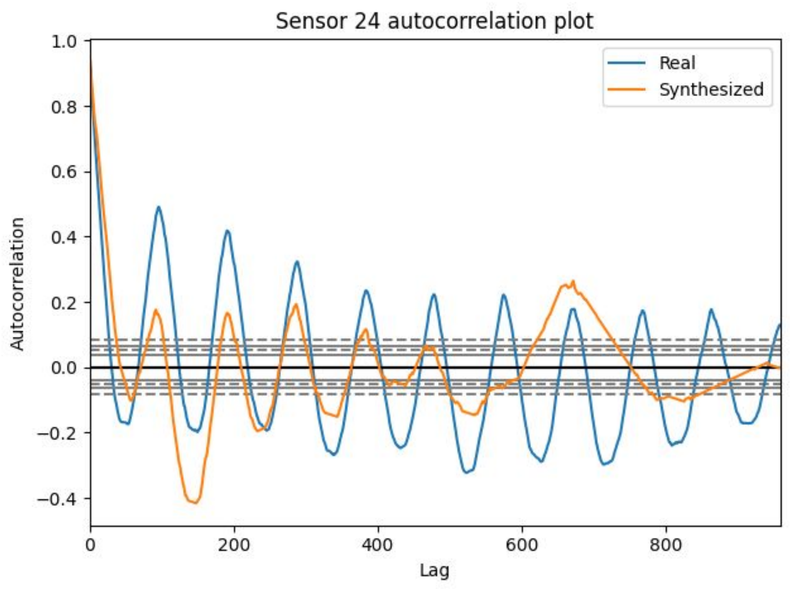
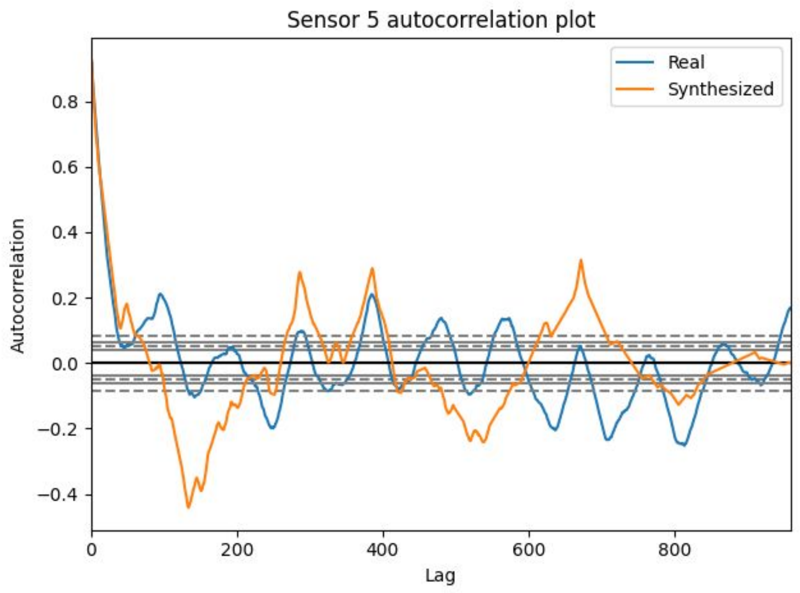


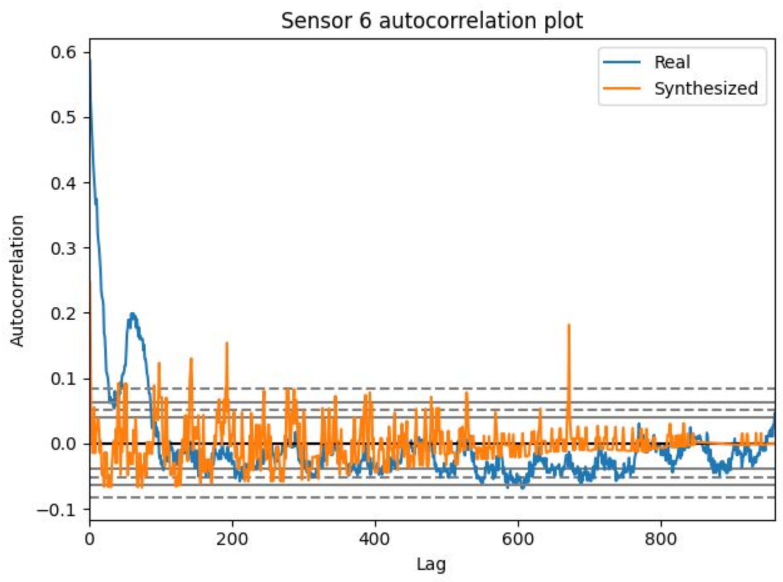
Figure 26 - Training set sensor values

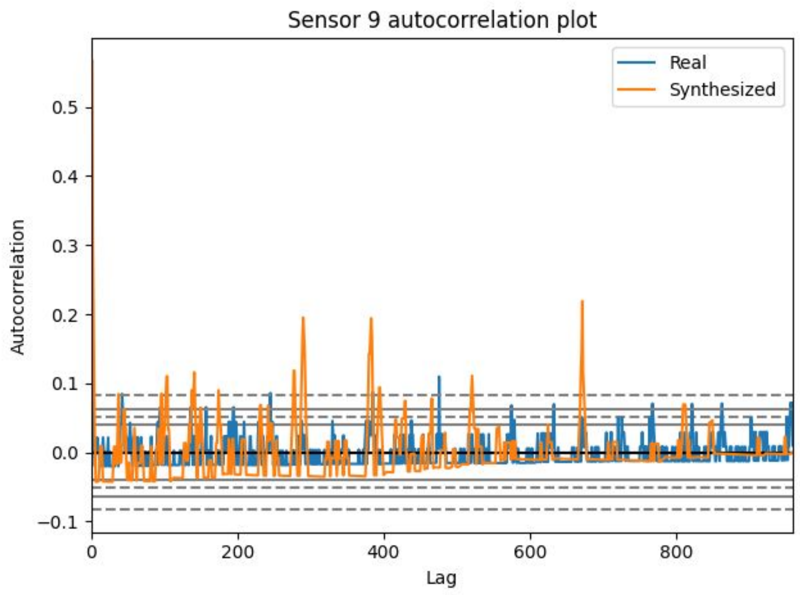


## Autocorrelation plots









*Figures 27-30 - autocorrelation plots of real and synthesized data*

### Entropy information

| **Sensor id** | **Sample entropy** | **Sample entropy 24 hours** | **Entropy 24 hours** | **Entropy** |
|---|---|---|---|---|
| *24 (real)* | 0.047 | 0.041 | 0.529 | 0.977 |
| *24 (synthesized)* | 0.041 | 0.028 | 0.467 | 0.975 |
| *5 (real)* | 0.060 | 0.081 | 0.508 | 0.997 |
| *5 (synthesized)* | 0.055 | 0.066 | 0.493 | 0.963 |
| *6 (real)* | 0.063 | 0.116 | 0.254 | 0.473 |
| *6 (synthesized)* | 0.068 | 0.088 | 0.226 | 0.341 |
| *9 (real)* | 0.034 | 0.057 | 0.088 | 0.139 |
| *9 (synthesized)* | 0.045 | 0.057 | 0.163 | 0.255 |

## Discussion

Visually, the synthesized sensor values look quite similar to the training set sensor values. In both the training and test data, sensor 6 has some periods where there are many activations and others where there are no activations for an extended time. Sensor 9 has very short activations, with a longer time between each activation. Sensor 24 and 6 follow a fixed pattern, although the synthesized data seems more irregular.

The mean synthesized values give an indication of the uncertainty of the prediction. The uncertainty for sensors 9 and 6 is very high - the expected value remains below 0.4 at all times. As expected, these sensors are more difficult to predict than the other sensors. Combining all simulations gives a much more smoothed out version of the sensor values - the same pattern seems to repeat itself every day. This is not the case in the training data - there are clearly some days where the house is probably vacant as there are no sensor activations at all on these days.

The  autocorrelation plots and entropy values also show that the quality of the synthetic data is not very high. Especially for sensors 6 and 9, the difference in entropy of the synthetic/real data is quite large. This indicates that there are either too many or too few activations. In theory, if the synthetic data accurately reflects the real sensor data, the autocorrelation plots would also be similar. The autocorrelation plot of sensor 24’s synthetic time series is quite similar to the training data. The accuracy of the prediction in the test data might indicate that the model has managed to learn to predict sensor 24's values quite accurately. The autocorrelation plots of sensors 9 and 6 do not have a clear pattern and are much more noisy. These sensors have fewer activations and shorter activation times which means they are more difficult to predict.

## Conclusion

10 days of data were synthesized using a single day of past data. Although the synthesized data looks visually similar, the autocorrelation plots and entropy metrics reveal that this is not quite the case.

Given more training data and more time to improve the neural network design, a more accurate model could probably be created which would reflect the real sensor values more accurately.

# Appendix

## Sensor values

### House 1

| id | name | activations | mean duration [s] | std duration [s] |
|---|---|---|---|---|
| 24 | Hall-Bedroom door | 118 | 10,750.23 | 28,754.70 |
| 5 | Hall-Toilet door | 180 | 5,460.41 | 16,856.88 |
| 6 | Hall-Bathroom door | 240 | 513.43 | 3,884.86 |
| 9 | Plates cupboard | 62 | 15.35 | 59.12 |
| 8 | Fridge | 140 | 4.87 | 6.12 |
| 1 | Microwave | 30 | 20,167.77 | 110,317.06 |
| 23 | Groceries Cupboard | 65 | 539.20 | 4,257.51 |
| 13 | Dishwasher | 19 | 163.68 | 216.96 |
| 18 | Pans Cupboard | 50 | 1,127.54 | 6,453.21 |
| 7 | Cups cupboard | 49 | 12.43 | 42.68 |
| 17 | Freezer | 36 | 7.67 | 5.85 |

### House 2

| id | name | activations | mean duration [s] | std duration [s] |
|---|---|---|---|---|
| 16 | badkamer klapdeur links | 229 | 1.00 | 0.00 |
| 38 | wasbak boven, flush | 142 | 1,062.58 | 5,914.16 |

| | | | | |
|---|---|---|---|---|
| 8 | toilet flush boven, flush | 43 | 33,003.72 | 37,098.14 |
| 35 | badkuip, pir | 1565 | 362.72 | 4,037.74 |
| 23 | kastje borden/kruiden,reed | 82 | 5,828.27 | 19,373.81 |
| 30 | koelkast, reed | 227 | 6,801.29 | 16,636.14 |
| 18 | bestek la, kwik sensor | 202 | 8.74 | 109.90 |
| 27 | kastje cups/bowl/tuna, reed | 33 | 37,150.73 | 48,789.37 |
| 7 | vriezer, reed | 22 | 29,681.32 | 57,970.11 |
| 20 | kastje pannen, reed | 18 | 71,579.44 | 123,856.38 |
| 21 | magnetron, reed | 26 | 19,562.88 | 58,120.92 |
| 22 | kastje restjesbakjes, reed | 6 | 146,222.00 | 312,273.91 |