CS 4993 Independent Study Report

Siyuan Shen
ss8rs@virginia.edu

I. Background

The goal of the independent study is to improve the accuracy, data collection and the testing process in the Detecting Abnormal Behaviors in Smart Home paper[1].

As IoT technology and smart home devices have become increasingly more popular, one interesting topic about home security is the detection of abnormal activities at home. Previous work in this field mostly focused on the detection of some specific and pre-defined abnormal behaviors, such as the activity of the elderly with dementia[2] and patients with illness[3]. Two potential problems were raised with these methods. First of all, a comprehensive model needs a large amount of labelled data that covers a wide range of abnormal behaviors for training. Secondly, as abnormal behavior is considered as a rare event, it is extremely difficult to collect sufficient samples in real life. In addition, the system needs to be adapted to different environments. Behaviors considered as normal in one environment might be considered as abnormal in other scenarios. Therefore, different sets of abnormal behavior data are required for each smart home scenario, which adds even more difficulty to the already challenging abnormal data collection process. Our research tries to solve the above problems through one-class classification, which means only the normal behavior data is needed for training. Compared with abnormal behavior data, the normal data is much more abundant and easier to collect through experiment. We can set up the sensors in a room, ensure no abnormal activities such as break-ins are occurred in the collection period, and then regard all the data collected as normal data.

II. Abnormal Data Simulation

Both the normal and abnormal data include a timestamp (measured in seconds), day of the week, door status, motion, acceleration and temperature. Even though the models used in the research are trained entirely based on normal data, abnormal behavior data are still needed for in the testing step to measure how well the models are performing. Apart from the original abnormal data simulated by other researchers in this project, I simulated one more abnormal data with more complexities and irregularities, hoping to test if the model can perform well

under more than one abnormal scenarios. The newly simulated abnormal dataset tries to improve two issues of the original abnormal data. In the original abnormal data, the average time between two events should be smaller, and the inactive status of acceleration and motion should occur more frequently.

The average time difference between two closest events (two consecutive samples) in the original abnormal data is 43.0 seconds, which is way much greater than the average 6.04 seconds in the normal dataset. According to the data from the Samsung SmartHome web console, we know that if we simulate an abnormal activity, such as opening the door consistently, or sudden break-in during mid-night, the sensors will record and update data more frequently during and after the abnormal activity. Thus, the time difference between two events in abnormal activity data would be smaller or at least not greater than that in the normal data. By looking at the distribution from the original abnormal data, it is safe to assume the time interval between each consecutive event follows a beta distribution. The new abnormal data is simulated with the beta distribution function $beta.pdf(y, a, b, loc, scale)$ in Scipy library in Python. The beta distribution follows the probability density function $f(x, a, b) = \frac{\Gamma(a+b)x^{a-1}(1-x)^{b-1}}{\Gamma(a)\Gamma(b)}$, where Γ is the gamma function. In the simulation, $a = 1.394$, $b = 9.54 * 10^{13}$. x is calculated by shifting and scaling y with $y = \frac{(x - loc)}{scale}$, with $loc = -0.068$ and $scale = 3.66 * 10^{14}$.

Abnormal activities tend to trigger the motion and acceleration sensor to change from 'inactive' to 'active'. Even though the acceleration and motion status are 'active' for most of the samples in the abnormal data, there are still certain number of 'inactive' status occur from time to time. Having more 'inactive' status could potentially make it harder to detect abnormality. Thus, in the new abnormal dataset, 'inactive' status is simulated such that they occur much more frequently. The time interval between two closest samples with 'inactive' status in either acceleration and motion is simulated with a beta distribution, with $\alpha = 0.571$, $\beta = 33.3$, $loc = 0.00285$, and $scale = 1413.7$

The newly simulated data is used to test the performance of the model under more complex scenarios. The new data is tested with the LSTM model mentioned below and it results in the same level of accuracy.

III. Long-Short Term Memory Model

The Long-Short Term Memory model (LSTM) is proposed in addition to the one-class Support Vector Machine Model and the Auto-encoder. LSTM works well for processing, classifying and making prediction for time series and data in a sequence. In order to fit the data into the LSTM model, the raw data is transformed from a shape of sample_size * num_of_features to sample_size * num_of_steps * num_of_features, where num_of_steps is the number of timestamps we trace back in one sample. The number of steps in our experiment is 20 and the num of features is 5. Each sample in the transformed data has a shape of 20 * 5. Suppose the training data is X_{train} and event at time t in the data is s_t (includes temperature, motion, and acceleration, etc.) For a sample at timestamp t, the independent variable (feature) x_t includes events from s_{t-1} to s_{t-20} and the dependent variable (true label) y_t is the values for event at time t, which is s_{t_true}.

The LSTM model is implemented with three layers. The first layer has 128 neurons and a ReLu activation function. The second layer has the same activation function but with 64 neurons. The output layer is a dense layer with 5 neurons. The output is an array with 5 values, each represents the predicted value for each feature.

The model is trained with the normal behavior data alone, with 80% of the samples as training data and 20% for validation. The model is trained to minimize the mean squared error (MSE) between the predicted values s_t from LSTM and s_{t_true}, the ground truth values for the sample at time t. The threshold is established by measuring the above mean squared error for each sample in the normal validation dataset such that it allows 5% of the normal behaviors wrongly predicted as abnormal. Therefore, the model allows us to set and control the false positive rate. During the prediction phase, events data from time s_{t-1} to s_{t-20}, where t is the current timestamp, are used as inputs to the LSTM model. Then the distance between the predicted output s_{t_pred} and the event detected at current timestamp t from the sensor s_{t_true} are calculated with mean squared error. If the MSE between the predicted and current true events are above the threshold, it indicates the event is deviated from the predicted output by the LSTM model, and thus we predict the event to be abnormal.

IV. Long-Short Term Memory Auto-encoder

The Long-Short Term Memory Auto-encoder combines the idea of the LSTM model and the auto-encoder. Here, the LSTM auto-encoder involves a compression and decompression stage. In the LSTM auto-encoder, each sample still has a shape of 20 x 5 (num_of_steps * num_of_features). In X_{train}, sample x_t, where t is the current time, contains events data from time s_t to s_{t-19}, instead of s_{t-1} to s_{t-20} in the previous LSTM model. Here, in the LSTM auto-encoder model, the target y_{train} is the same as x_{train}, which means we are using the model to reproduce the input. The model will try its best to reproduce the its input as the output. The LSTM auto-encoder takes a 2-D array with size 20 x 5 as input. The input first goes through a two-layer encoder with 64 and 32 neurons respectively and ReLu activation function. Then, there is a two-layer decoder that with 32 and 64 neurons and ReLu activation function. The final output reproduced by the LSTM auto-encoder model is still a 2-D array with size 20 x 5.

The model is still trained with the normal behavior data, with 80% of the samples for training and 20% for validation. The mean squared error between the input and the output is calculated, so that it can be used as a threshold for prediction. Similar to the LSTM model above, the threshold on the MSE is adjusted such that the false positive rate is set to 5%. During the real-time detection phase for testing, a 2-D array that contains all five statuses from s_t to s_{t-19} are used as input to the model. If the mean squared error between the input and the output exceeds the threshold, the current event at t is then considered as abnormal.

V. Model Performance

The models in this project are tested on a dataset with abnormal behaviors only. The table below shows the performance of the One-class SVM, Auto-encoder, LSTM and LSTM auto-encoder. The abnormality accuracy is measured based on the abnormal behavior data only. From the table, the two LSTM models outperformed the One-class SVM and auto-encoder.

Table 1: Comparison of Model Performance

Model	Overall Accuracy	Abnormality Accuracy	False-positive Rate
One Class SVM	83.4%	98.6%	31.8%
Auto-encoder	83.7%	72.3%	5.1%

LSTM	91.0%	83.1%	5.0%
LSTM auto-encoder	96.7%	100%	5.0%

VI. Testing Data Simulation

In a realistic smart home scenario, abnormal activities last only a short period of time compared with the length of the entire data and they occur in between normal activities. In order to further testing the performance of the models under scenarios closer to real-life, a testing dataset with a mixture of normal and abnormal activities are generated. The normal activities were real-life data collected by the Samsung SmartHome sensors. The mixture data is generated by randomly picking intervals in the normal data and replaced by simulated abnormal behavior data. Once a set of timestamps are picked and replaced with abnormal data, it won't be picked again to ensure the integrity of the inserted data. The generated mixture data was tested both with the LSTM and LSTM auto-encoder. Nevertheless, the resulting accuracy is not promising, partly because the proportion of abnormal data in the dataset are way much higher than what would happen in real life. Therefore, more future work is needed in simulating the mixture testing data.

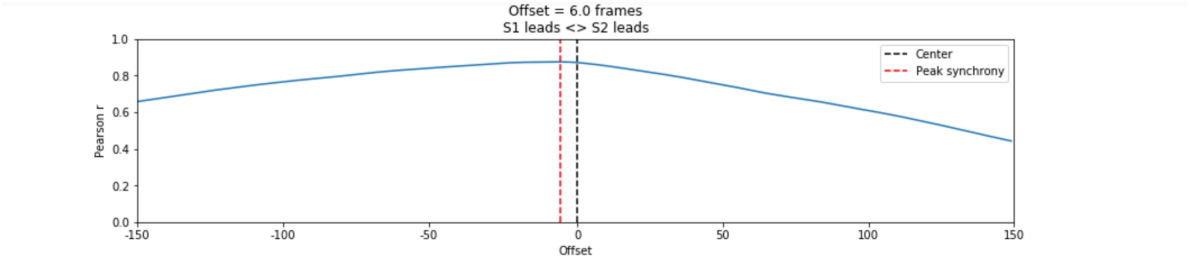
VII. Detecting Malicious Attacker Devices

Malicious attacker device is able to send data that misleads the SmartHome sensors. Here in our experiment, malicious temperature features were simulated and combined with features from SmartHome sensor to confuse the detection. The attacker device has no access to the environment inside the room in which the sensors are located. Thus, the readings from the attacker device are not synchronized with the readings from the SmartHome sensors. We have multiple sensors that records the temperatures in the room. Due to differences in the type, timing and location of the sensors, the readings are not exactly the same. However, the temperature features from the three SmartHome sensors are in mostly synchronized. Here, we try to detect the features coming from an attacker device by utilizing this property.
The detection is made possible by measuring the cross validation on two time series data. We shifted one of the time series incrementally so that there is a lag between the two time series, and then measure the cross correlation. We picked the lag with the highest correlation. With that, we can know if the two time series are synchronized. A threshold is picked such that when the offset is greater than the threshold, the time series is considered as coming from a malicious device. In this experiment, the threshold is determined to be 60 based on

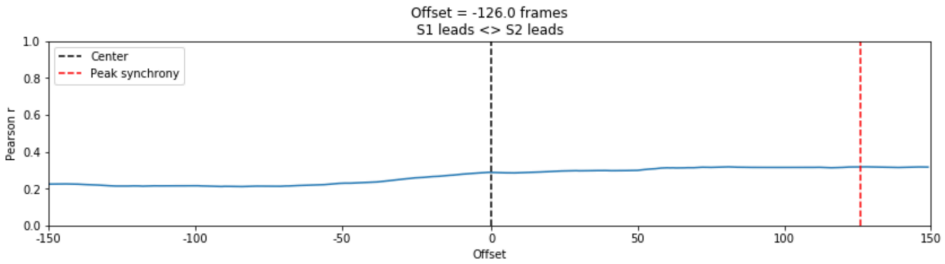
observation. In the future, the threshold can be set to according to the tolerance on false positive rate.

The reason why the cross-correlation is measured across different time lags by shifting one time series data with another, instead of simply calculating the Pearson correlation between the two series directly, is that different sensors in the same room might never be synced perfectly at the same time. There might be slight difference in the time in which each sensor in the room detects and updates events due to the relative location of the sensor in room and the type of the sensor, etc.

Below are two images showing that the model is able to distinguish a dataset with two time series readings measured by sensors (real sensor data vs real sensor data) from a dataset with one time series as sensor readings and another from a potential attacker device (real sensor data vs simulated malicious data). The models correctly predict the first to be not malicious and the second one as malicious.



The first graph is a comparison between two temperature features from the sensors. As both of them come from the smart home sensors, they achieved their highest correlation of around 0.8 at an offset of 2 seconds. It is predicted to be not malicious based on the relatively small offset and a strong correlation across different lags.



On the other hand, the second graph is the result between a temperature time series from the sensors and a simulated malicious time series. As the malicious attacker device has no access to the information inside the room in which the sensors are located, the readings from the attacker device are not synchronized with other sensors in the room. The model gives a large offset, and a weak correlation across all lags, which correctly predicts the sensor data to be malicious.

VIII. Conslusion

The independent study proposed two models that improved the abnormal behavior detection accuracy. In addition, three more testing datasets were generated to test the models in more complex smart home scenarios. The study also illustrates a simple approach to determine if two sequences of readings are from the same environment. The future work will be focused on the sensor fusion and malicious attacker device detection by transforming the time series data into the frequency domain.

[1] Y. Yu, C. Li, M. Jonas, C Ma, and F. Shezan. Detecting Abnormal Behaviors in Smart Home

[2] D. Arifoglu and A. Bouchachia. Activity recognition and abnor- mal behaviour detection with recurrent neural networks. *Procedia Computer Science*, 110:86–93, 2017.

[3] A. C. Tran, S. Marsland, J. Dietrich, H. W. Guesgen, and P. Lyons. Use cases for abnormal behaviour detection in smart homes. In *International Conference on Smart Homes and Health Telematics*, pages 144–151. Springer, 2010.