Detecting Abnormal Behaviors in Smart Home

Yonghua Yu
University of Virginia
yy5uu@virginia.edu

Changyang Li
University of Virginia
cl5xf@virginia.edu

Mainuddin Ahmad Jonas
University of Virginia
maj2bh@virginia.edu

Chuanhao Ma
University of Virginia
cm9mb@virginia.edu

Faysal Hossain Shezan
University of Virginia
fs5ve@virginia.edu

Siyuan Shen
University of Virginia
ss8rs@virginia.edu

Peng Gao
University of California, Berkeley
penggao@berkeley.edu

Yuan Tian
University of Virginia
yuant@virginia.edu

***Abstract*—Recent popularity of smart home devices, due to the widespread use of Internet of Things (IoT) technology, has enabled infinite possibilities to provide people with more convenient, safe and healthy life experiences. Among various ideas and applications in smart home environments, the problem of abnormal behaviors detection arises people's attentions and concerns. In our work, we propose to achieve anomaly detection as a one-class classification problem. In particular, we use two machine learning models, which are Autoencoder and one-class Support Vector Machines, to learn the normal behavior patterns and then detect any behavior that deviates from the learned normality. We further use Samsung SmartThings, an open platform for smart homes and the consumer IoT, to build a real-time detection and notification mechanism, and evaluate effectiveness of our model in both offline and online detection scenarios.**

I. Introduction

Home automation is building automation for a smart home environment by integrating intelligent IoT sensors and devices. It typically connects distributed sensors and devices to a central hub, collects all data and performs automated controls. Smart home systems usually include multiple sensors monitoring different types of data like temperature, lightness and motion in the house, and are used in some applications and research projects to monitor and analyze home status, aiming to enhance home security. One interesting and challenging topic about home security using smart home devices is human behaviors detection. Especially, we address the problem of abnormal behaviors detection in this paper, because this would be vital to defend any malicious behaviors at home and guarantee security. There are three main difficulties to this problem: Firstly, how should we define abnormal behaviors? Previous works usually focus on one specific type of abnormality, like Tran et al. [12] focus on inhabitant's illness and unexpected behaviour, and Arifoglu et al. [1] care about a few behaviors of elderly people with dementia. As a result, these systems are only able to detect limited numbers of abnormal behaviors that are predefined, and a more comprehensive system might need great efforts to define enough more abnormality patterns. Secondly, how should we collect and process data to train the detection models? According to our best knowledge, state-of-art approaches on this task are developed based on data-driven methods, specifically supervised machine learning methods. Therefore, huge amount of data should be annotated, especially when a lot of abnormal behaviors are covered according to the first difficulty. Thirdly, how should these systems be applied in different smart home scenarios? As types, numbers and set locations of smart home sensors vary a lot, we might hardly train a general model that fits every home scenario. However, to train a model that works only under a specific home settings would be time-consuming, considering the previous two difficulties discussed.

To handle these difficulties, we propose to achieve the abnormal behaviors detection task as a one-class classification problem [8]. In one-class classification, the classifier tries to identify objects of a specific class among all samples by learning from a training set containing only the samples of that class, and in our work, all data captured in day-life is regarded normal for training. Our aim is to train a model that could learn the normal user patterns automatically, and regard detected patterns that are apparently different from the learned user patterns abnormal in detection process. Corresponding to the difficulties discussed before, our method has the following advantages: Firstly, while our method should have the ability to detect all abnormality, we do not need to specify the types of abnormal behaviors. Secondly, we do not need data annotations, as our training set should be full with normal data. Thirdly, this model could be easily applied to any smart home scenarios and it will do self-training automatically.

In our work, we develop our abnormal behaviors detector model, which is a one-class classifier, using two different methods. We explore the possibility of using

978-1-7281-4121-3/19/$31.00 ©2019 IEEE
DOI 10.1109/MASSW.2019.00014

Autoencoder, which is a supervised-learning method. However, we converting it to the one-class classification scenario by training it with normal data only, and the result is that only normal patterns could be successfully reconstructed by the decoder in the model. We also use one-class Support Vector Machines (one-class SVM), which learns the boundary of normal user patterns instead of maximum margin of two types of data. Apart from our model, which is the key element in our work, we developed an smart home application that does abnormal behaviors detection in real-time on Samsung SmartThings platform. The main contributions of our work is three-fold:

- Presenting a novel approach that solves abnormal behaviors detection in smart home as a one-class classification problem, and implementing our model with two methods.
- Evaluating the effectiveness of our approach for abnormality detection.
- Developing a smart home application for real-time detection on Samsung SmartThings platform.

II. Related Work

Human Behaviors Detection. Some early studies use logic-based models for behaviour analysis, such as the Use Cases based model presented by Tran et al. [12]. This method introduces the idea of Use Cases in Software Engineering, pre-defining a set of abnormal scenarios and analyzing within the defined logic models. While logic-based models might be explainable, they could hardly handle events outside of the defined scenario set. However, scenarios nowadays are often presented by multiple types of sensors jointly in a smart home, and it becomes a main difficulty for the logic-based models to deal with the huge solution space of behaviour detection problem. Later, increasing availability of data allows the problem to be solved by data-driven methods, and in particular more researchers prefer to apply machine learning approaches for human behaviour detection. One significant point of applying machine learning methods is that detection models could automatically learn user policies encoded in data from connected sensors and devices, and are able to handle various scenarios. For example, some studies take use of Support Vector Machine (SVM) for abnormal behaviour detection [4], [3]. More recently, great developments of deep learning methods bring more possibilities into the detection task. For example, there has been growing interest in applying deep convolutional neural network (CNN) [15], [13]. Some other studies use Long Short-Term Memory network (LSTM) and Recurrent Neural Network (RNN) [2], [1] to address the temporal relationship. Though deep learning methods are currently taking the lead in performances for abnormal behaviour detection, one major limitation is that huge amount of annotated data is necessary for training. To avoid this, we use one-class classification approach to solve the problem.

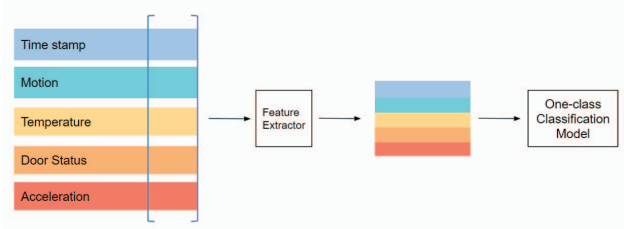

Fig. 1. An overview of our approach.

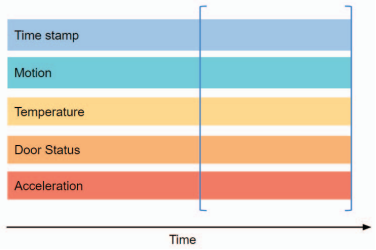

Fig. 2. The detection window used to extract data within a period of time. In our work, we set the window size 100 seconds and the sliding step 1 second.

One-class Classification. The one-class classification problem [5] is different from the conventional binary/multi-class classification problem, as the negative class is either not present or not properly sampled. Moya et al. [8] first present one-class classification approach for target recognition, and later studies use this concept in various application scenarios, like text classification [7], [14], handwritten digit recognition [11] and machine fault detection [10]. We address that one-class classification suits our application scenario well, which is abnormal behaviors detection, as one-class classification approach could learn normal user patterns automatically without manually labelling training samples. In our work, we explore two types of one-class classification approaches – one-class SVM [9] and autoencoders [6]. The first one is a popular classical machine learning approach for this problem, while the latter is a neural network-based approach that has gained traction in research community in recent years.

III. Approach

In this section, we discuss details of our one-class classification approach for abnormal behavior detection. We first discuss data and corresponding features used in our work, and then give out two designs of our classifier, which are Autoencoder and one-class SVM separately. In our approach, we organize all types of data chronologically, and use a fixed length detection

window that slides along the time axis to extract a batch of data for detection each time. Features of data in current window are extracted and passed into our one-class classification model to judge if abnormality is occurring. Figure 1 illustrates this idea.

A. Data

To train a general abnormal behaviors detector that could be easily applied to various smart home scenarios, we do not train our model with public dataset. Instead, we train and test the detector using data collected from our experimental scenario directly. All of the collected data is regarded normal and used as user patterns for training, and we synthesize fake abnormal data in offline abnormality detection experiment in Section V. We align all collected data along time axis, and set a detection window that keeps sliding, which is shown in Figure 2. We set the window size 100 seconds, and the sliding step is 1 second along the time axis. Each detection window is a basic raw input to our detection approach, and this is a key design in our work to handle temporal relationships because an abnormal behavior might last for a period of time rather than occur for a second. We care about the following types of data:

1) *Time stamp*: Time in a day, and we ignore the date as it doesn't depend much on date.
2) *Motion*: Whether motions are detected.
3) *Temperature*: Monitored temperature.
4) *Door Status*: Whether the door is close or open.
5) *Acceleration*: Whether the door is moving with an acceleration or not.

B. Feature

Our feature extractor work according to our defined rules. We extract some statistical results as features considering the sparsity and diversity of raw data, which make the model training difficult. We have the following definitions of features corresponding to data types:

1) *Time stamp*: We divide one day into 48 zones, and the time stamp feature is the index of time zone the target detection window belongs to.
2) *Motion*: Count of how long the motion status is active in the detection window.
3) *Temperature changes*: Count of times temperature changes in the detection window.
4) *Average temperature*: The average temperature in the detection window.
5) *Door Status*: Count of how long the door status is open in the detection window.
6) *Acceleration*: Count of how long the acceleration status is active in the detection window.

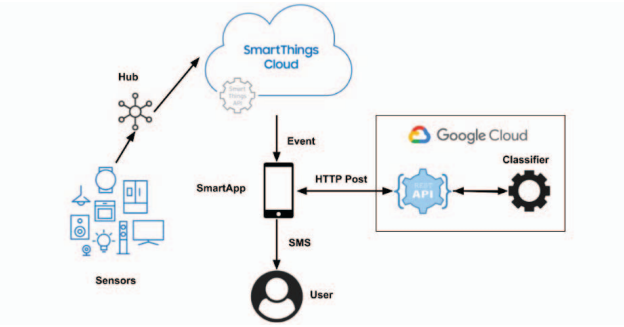

Fig. 3. Framework of smart home application.

C. Model

In our work, we have two versions of abnormal behaviors detection models, which are Autoencoder and one-class SVM. Note that while the method of Autoencoder is a supervised learning method, we convert and apply it into the one-class classification scenario by training it with normal user patterns only.

Autoencoder. Autoencoders are a type of neural network that reproduces its input to its output. The main idea behind autoencoders is to convert a large number of feature representation into a smaller number of feature representation, and then try to get back the original representation from that. Hence, there is a compression, and a decompression step inside the autoencoder model. The compression is done through an *encoder*, and the decompression is done through a *decoder*. Autoencoders have been found to be useful for mainly two types of tasks – denoising and manifold learning. In our project, we make use of the latter. By converting the original feature space to a smaller representation, the model is forced to learn the most salient features of the dataset, and ignore the less important features and noises. As a byproduct of this, the model also learns the hidden manifold of the training data. This last part is the key to our project. We train our model only on normal data, and in the process the model learns the manifold of normal data. At test time, when the model encounters an abnormal data, which lies off the manifold, it fails to reproduce the input. Using this property, we can detect abnormal behavior even without using any abnormal training data.

One-class SVM. Tradition Support Vector Machines (SVM) projects all data points to a higher dimensional space, and tries to find a maximum margin between two types of data. However, as one-class SVM [9] is trained with only data of one type, it learns the boundary of the training data. While a good boundary should be close to training samples in original space, a one-class SVM model tries to separates samples from the origin in feature space and maximizes the distance from

the hyperplane to the origin. We feed all training data, which represents normal user patterns according to our definition, into the one-class SVM model for training. Therefore, the model learns user patterns automatically, and should have the ability to find abnormal samples that are out of the learned boundary.

IV. Smart Home Application

We developed a smart home application one Samsung SmartThings platform (SmartApp), and the framework is shown in Figure 3. For a complete real-time detection and notification framework in smart home situation, there are mainly four important modules. They work together to support the whole mechanism. The Samsung SmartThings platform provides a API for users' to develop their own SmartApp. It is a type of plugin installed in the official SmartThings mobile application. It is very similar to the Alexa Skill in the Amazon Echo Dot. In the web console we mentioned above, it provides an IDE, as well as some classic templates to help users develop SmartApps.

The SmartApp is based on Groovy. The platform has packaged many libraries, and it exposes several interface to the developers. Developers use the subscriptions to receive the events or states of the IoT devices in real time, and develop the handler to process the events. Developers can test the SmartApp in virtual or real environment, and then publish it to the SmartApp marketplace. In this way, users can install it in its mobile SmartThings app and choose the proper IoT devices as the data source.

A. Devices

We use Samsung SmartThings as the IoT platform. Samsung SmartThings is a smart home platform. In the provided pack, there are one central hub and four sensors. The hub connects the basic IoT devices like sensors and lights wirelessly, and it can control and monitor these devices through the SmartApp installed in host's mobile phone. The given sensors includes two multipurpose sensor, one motion sensor and one outlet, which can monitor door and windows, temperature, vibration, motion and electronics. Another reason we choose SmartThings platform is that it provides a web console to easily manage the account. In the web console, we can explore the devices, hubs, SmartApps connected to the account. Through the web console, we can explore the devices' status and events history, including the temperature change and open or close of door and windows. In this way, if there is no event in some time period, it means there is no change.

B. IoT Detection

First of all, the framework needs input information. In smart home environment, IoT devices is responsible for the collecting data task. Currently, consumer IoT devices contain many different types, including motion sensor, temperature sensor and acceleration sensor. Using these IoT devices, we can monitor some physical attributes in the home in real time.

metric	overall accuracy	abnormality accuracy	false positive rate
One-class SVM	83.4%	98.6%	31.8%
Autoencoder	83.7%	72.3%	5.1%

TABLE I
NUMERICAL RESULTS OF MODEL PERFORMANCES ON TEST DATASET.

C. Cloud Server

In case that we want to build a real-time mechanism, the collecting data has to be processed in real time, a web server is necessary. Since most IoT devices have connection to the Internet, we can forward the data to the server. In the server, the processing application can deal with the data and judge whether there are some abnormal behaviors. Due to the rapid development of cloud service, the server can be easily deployed on the AWS or Google Cloud.

D. Classifier

The classifier is the core part of the cloud processing application. It can be trained in advance and then integrated with the API and deployed together into the cloud. Or, it can be directly deployed in the cloud at first and trained automatically. We discuss technical details about classifier model in Section III.

E. Notification

When the classifier produce its judgment, if there are something abnormal, the host have to be notified immediately. Therefore, we need a real-time notification mechanism. Fortunately, there are many methods to achieve this goal. For instance, SMS, mobile application push, or direct phone call. There are plenty of free web services to do that.

V. Experiment

In this section, we first discuss details about our dataset. We also discuss an offline experiment that evaluates our models performances on test dataset for abnormal behaviors detection, and an online experiment using our smart home application for real-time abnormality detection.

A. Data Preparing

To prepare data the model needs to train and test on, we used techniques including web crawling to get real data collected by IoT devices, also to create fake data to simulate abnormal situations for test. We built a training dataset with 863,901 normal samples, and a test dataset with 13,918 samples, within which 6,961 are abnormal.

Real Data Collection. As SmartThings platform provides a web console which shows all events recorded by IoT devices, using a web crawling technique is an easy way to collect those data. We selected Scrapy, a web crawling framework in Python which we are familiar with, for this job. After analyzing the web structure of the pages showing data, we were able to write scripts with Scrapy to extract events out of given web page and store them into .csv files for further usage. The data is used for training our models.

Fake Data Generation. In our assumption, all the possible data collected before should be seen as normal data. Also, it's hard to perform a lot of abnormal behavior to collect enough data to test. Thus after discussion and research on patterns of abnormal behaviors, we came up with several patterns that should be seen as abnormal in most situations. The categories include break-in in midnight, abnormal temperature changes, and frequent getting in and out of the room. To make sure the behaviors are close to reality, each behavior is formed by a series of individual and continuous events. Also, all of the generated behaviors are randomly inserted into normal data. The data is used to test our models.

B. Offline Experiment

We conducted the offline experiments on both Autoencoder and One-class SVM models on the test dataset, and numerical results are listed in Table I. While Performances of the two models vary, we talk about their advantages and disadvantages in the following discussions.

Autoencoder The autoencoder model was trained only on normal dataset. The model had a Densely connected input layer with dimension 5, followed by another Dense layer as encoder with dimension 2, followed by a Dense decoder with dimension 5. We used 80% of the training data for training, and remaining 20% for validating and choosing threshold. We used the Adam optimizer and mean squared error reconstruction error for training the model. Once we trained the model, we used the validation data for setting a threshold on the reconstruction error, such that our false positive rate is below 5%. Our actual classifier uses the trained model and the threshold to make decisions on whether a particular sample is normal or abnormal. In our experiments with simulated abnormal data, we were able to achieve detection rate of 72% when we set the false positive rate to be 5%.

One-class SVM According to results shown in Table I, our One-class SVM achieves impressive abnormality detection accuracy, which is 98.6%. However, we argue that the model could be unstable in normal scenarios as the false positive rate is high. This result suggests that while the model could detect most of abnormal behaviors, it also report false abnormality frequently. In comparison, though our Autoencoder model could not reach abnormal behaviors detection accuracy as high as One-class SVM does, the false positive rate is controllable.

C. Online Experiment

Besides the offline model evaluation, we also carried out online experiments on the real situation in one of teammates' home. We set up the SmartThings sensors and hub in his home, developed a SmartApp plugin installed in his mobile phone, and deployed an API in Google cloud to process. Our SmartApp subscribes all the events from sensors, including change of temperature, acceleration, motion and door status. Then we use http post request to send the changing data to the API deployed in Google cloud. In the cloud, we have the well trained classifier to judge whether the event indicates an abnormal behavior, send the response back to the SmartApp. The SmartApp has the API to directly sending SMS to the given phone number. We used that as the notification method to warn the host if the event is judged as abnormal one. In order to test the whole framework, we carried out some abnormal behaviors deliberately in the experiment room. For example, we open and close the door multiple times in a short period of time at midnight, the time that users usually in his sleep. In a few seconds, the mobile phone successfully received the abnormal warning. Furthermore, we built a small list of some classic abnormal behaviors, just like what we faked in the offline experiment. We tested all of them in the real situation using the One-class SVM classifier model. The model detected all the abnormal behaviors. However, it had some wrong judgment on the normal behaviors, which indicates false positive.

VI. Discussion

In this project, we have two main contributions. First, we implemented two machine learning models as novel approach to detect abnormal behavior in smart homes, and evaluated the performances. Based on the models, we have developed a Samsung Smartthings Application for real-time abnormal behaviors detection. The models are effective with acceptable performance, but they have also limitations. The One-class SVM approach has a high accuracy with also high false positive rate, which means it tends to consider some normal behaviors as abnormal. And the Autoencoder model has a lower accuracy but

also lower false positive rate. In the future, we plan to improve our data collection and simulated data generation methods to improve the quality of the training and testing data. In particular, we would like to add more features to our dataset which are currently not available through the web console API. Adding more features will help us train better models, and learn the manifold of the data more accurately. As for generating simulated abnormal data, we would like to test more scenarios of abnormal data other than break-in, and see if our model can be extended to handle those scenarios. We also plan to make the SmartApp more usable and configurable. Currently, the classifier is trained in advance by collected data. All the work is done manually. Therefore, the SmartApp can be only applied to the specific user and specific situation. In the future, the whole framework can be general and automatic. For a new user installed the SmartApp, we first collect all the devices data for a period without any other behavior, use the data to primarily train the classifier model. And then activate the warning function based on the classifier judgment. Meanwhile, the classifier can be continually improved by collecting more data as the user use the SmartApp.

VII. Conclusion

In this project, we built an end-to-end system for detecting abnormal behavior in a smart home system. Through this project we faced a few challenges that are different from anomaly detection in computer systems. Our first-step investigation will help researchers to build anomaly detection systems with physical information.

References

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

[2] W.-H. Chen, C. A. B. Baca, and C.-H. Tou. Lstm-rnns combined with scene information for human activity recognition. In *e-Health Networking, Applications and Services (Healthcom), 2017 IEEE 19th International Conference on*, pages 1–6, 2017.

[3] B. Huang, G. Tian, H. Wu, and F. Zhou. A method of abnormal habits recognition in intelligent space. *Engineering Applications of Artificial Intelligence*, 29:125–133, 2014.

[4] V. R. Jakkula and D. J. Cook. Detecting anomalous sensor events in smart home data for enhancing the living experience. *Artificial intelligence and smarter living*, 11(201):1, 2011.

[5] S. S. Khan and M. G. Madden. A survey of recent trends in one class classification. In *Irish conference on artificial intelligence and cognitive science*, pages 188–197. Springer, 2009.

[6] C.-Y. Liou, J.-C. Huang, and W.-C. Yang. Modeling word perception using the elman network. *Neurocomputing*, 71(16):3150 – 3157, 2008. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).

[7] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394. Citeseer, 2002.

[8] M. M. Moya, M. W. Koch, and L. D. Hostetler. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93, 1993.

[9] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[10] H. J. Shin, D.-H. Eom, and S.-S. Kim. One-class support vector machinesan application in machine fault detection and classification. *Computers & Industrial Engineering*, 48(2):395–408, 2005.

[11] D. M. J. Tax. One-class classification: concept-learning in the absence of counter-examples [ph. d. thesis]. *Delft University of Technology*, 2001.

[12] 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.

[13] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001, 2015.

[14] H. Yu, J. Han, and K.-C. Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[15] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 197–205. IEEE, 2014.