FLEAD: Online Frequency Likelihood Estimation Anomaly Detection for Mobile Sensing

Viet Duc Le
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
v.d.le@utwente.nl

Hans Scholten
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
hans.scholten@utwente.nl

Paul Havinga
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
p.j.m.havinga@utwente.nl

UbiComp '13, September 08 - 12 2013, Zurich, Switzerland.

ACM 978-1-4503-2215-7/13/09...$15.00.

http://dx.doi.org/10.1145/2494091.2499774

Abstract
With the rise of smartphone platforms, adaptive sensing becomes an predominant key to overcome intricate constraints such as smartphone's capabilities and dynamic data. One way to do this is estimating the event probability based on anomaly detection to invoke heavy processes, such as switching on more sensors or retrieving information. However, most conventional anomaly detection methods are power hungry and computation consuming. This paper proposes a new online anomaly detection algorithm by capturing the likelihood of frequency histogram given features extracted from a stream of measurements from sensors of multiple smartphones. The algorithm then estimates the mixed density probability of anomalies. By doing so, the algorithm is lightweight and energy efficient, which underpins large scale mobile sensing applications. Experimental results run on Android phones are consistent with our theoretical analysis.

Author Keywords
Anomaly detection, outlier detection, energy efficient, mobile sensing, mobile platforms

ACM Classification Keywords
H.2.8 [Database Management]: Database Applications-Data mining.

Introduction

Modern smartphones enable researchers to utilize them as mobile sensor nodes to develop low-cost, steady, reliable and scalable sensing systems [1–3]. However, resource constraints and intermittent connectivity of smart phones make developing sensing systems challenging, particularly in dynamic environments and large-scale networks. We realized that being able to spot starting and ending moments of events, without knowing what the events are, can be used to reduce power consumption in many cases. For example, an adaptive sampling mechanism switches on a few sensors to monitor environments at low sampling rates. Knowing that an interesting event is about to happen, the algorithm invokes extra sensors or increases sampling rates to collect more information. A transition from an activity to another can be used to trigger the activity recognition process. Once the new activity is recognized, the recognition process can stop to save energy until the next transition is detected. Determining the probability an event might be happening is a dominant key in sensing with mobile platforms. Especially in public safety applications, interesting events, such as a fire or a car accident, are rare events. Therefore, the applications can be in sleep mode most of the time.

To detect the transition of the activities, anomaly detection can be applied on an online stream of measurements. Anomalies are described in different ways by different researchers. Herein we consider anomalies as patterns in data that have low density probability in a model built from historical data. However, existing anomaly detection algorithms such as Support Vector Machine, Multivariate Gaussian Model, Cross Entropy, Autoregressive Model, and Linear Regression require either numerous training samples or high computation to estimate model parameters, which are unrealistic in large-scale and spare mobile networks. There are also online training variants of anomaly detections. However, gradually learning parameters will result in considerable latency in dynamic environments. In addition, most conventional anomaly detections require heavy computation. Even though smartphones are getting more powerful, the battery's capacity is still very limited. Therefore, energy comsumption is still a very important issue in mobile platforms.

To this end, we developed an online algorithm estimating event probability given data from various sensors in highly dynamic contexts (e.g. public safety applications with smartphones). We named the algorithm FLEAD (Frequency Likelihood Estimation for Anomaly Detection). By elaborating the traditional frequency histogram, we obtained a good estimator to infer the density probability of data for testing events. This approach does not need an off-line training phase and performs well even with a small number of prior samples.

Sensor data are split into time windows based on predefined granularity. Features are extracted from the measurements based on granularity. FLEAD estimates the density probability of historical features based on an elaborated frequency histogram. Unlike other approaches using KL divergence to detect anomalies, we propose a mechanism to estimate the density probability of data using the sum of the frequencies at the least and most significant bins. Moreover, by taking the absolute value of the features, anomalies will fall into either the most left or right bin. This makes it easy to compute the event probability (anomaly probability). Another advantage of this approach is easily finding a stable threshold to detect anomalies by tuning the F_1-score, known from statistics. Experiment results with features extracted by taking

means of measurements that fall outside of the confidence intervals are consistent with our theory.

The paper is organized as follows. In the next section we discuss related work. Afterwards, we present the proposed algorithm in four sections: problem formulation, frequency likelihood estimation, FLEAD's pseudo code and complexity. Then, naive empirical experiments are carried out with mobile platforms. Finally, we end this paper with conclusion and future work.

Related Work

Histogram-based anomaly detection, the simplest nonparametric statistical method, constructs a profile of normal samples. Using a histogram is a well-known approach in various sensing applications, such as intrusion detection [4], fraud detection [5]. There are two steps involved in conventional histogram based detection. The first step is building a profile given normal data. The second step checks if test data fall in any of the bins. If not, the test data are anomalous. An alternative method is measuring the frequency of the bin in which data fall. A key challenge for conventional techniques is to determine an optimal size of the bins, which is a tradeoff between low false positive and low false negative. Sensing dynamic environments using mobile platforms raises another challenge that is the test data usually fall out of the learning profile even it is normal. Moreover, the frequency of the bin in which anomalous data fall may be as high as other bins of normal data.

There are also numerous algorithms for anomaly detection, which are well reported in two surveys of Chandola et al. [6, 7]. In this limited space, we only discuss some well-known algorithms. Versions of Support Vector Machine were implemented for one-class classification for outlier detection using Hamming distances [8]. This approach can deal with a small number of samples but requires high computation. Clustering Based [9] classifies thousands of text reports into a number of bins. Data falling into the class with smallest size is anomalous. A naive method to detect anomalies using Multivariate Gaussian Distribution was taught in the Machine Learning class by Andrew Ng. [10]. This technique is lightweight and simple but very effective. However, it also requires a large number of samples and computation is still high when there are numerous sensors, which happens quite often in smartphone crowdsourcing. In general, detecting anomalies from an online stream in a small time windows (say from 10 to 50 samples) is still an intricate research topic for mobile sensing platforms.

Problem Formulation

Consider a set of n measurement channels (e.g. light, ambient temperature, acceleration force along the x axis, y axis, z axis, and so on.). For an arbitrary sequential granularity k, x_i^k denotes feature extracted from m_i measurements $\delta_i^k = \{\delta_i^{1,k}, \delta_i^{2,k}, \ldots, \delta_i^{m_i,k}\}$ obtained from channel i. Features can be extracted by mean, fourier transform, wavelet, etc. Given a dataset of l previously consecutive features $\mathcal{X}^{t-1} = \{x^{t-1-l}, x^{t-l}, \ldots, x^{t-1}\}$ and the current granularity features x^t, the problem is to detect if x^t is anomalous. In other words, given sample data $\mathcal{X}^{t-1}$, we have to find the statistical density model probability $p(x)$. Then if

$$\begin{cases} p(x^t) \geq \epsilon, & x^t \text{ is normal} \\ p(x^t) < \epsilon, & x^t \text{ is anomalous} \end{cases}, \tag{1}$$

where ϵ is a threshold, which can be chosen in advance or tuned by maximizing the F1-score.

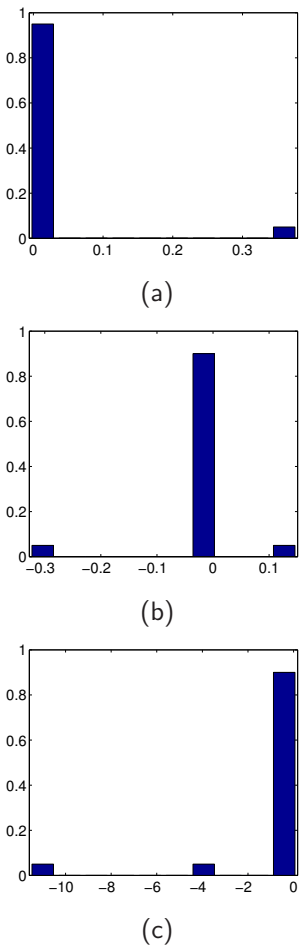

Figure 1: Histograms with outliers.

Frequency Likelihood Estimation

We propose a lightweight technique, named as Frequency Likelihood Estimator (FLE), to detect anomalies given a dataset. The term "frequency likelihood" comes from that the technique elaborates the frequency histogram of consecutive samples. We remark that using frequency histogram to estimate the probability density function of underlining variable such as Equation 1 is not a new idea. However, using KL divergence to detect anomalous data as in conventional methods is not suitable for highly dynamic histograms. In particular, a small amount of samples measured in a dynamic environment typically is not normal distributed and the mean value jumps broadly from time to time, especially when the length of historical data is short. This dynamic also makes it hard to determine a fixed threshold. Indeed, our new method is able to deal with such issue.

Given dataset $\mathcal{X}^t$ of $l+1$ samples, FLE first counts the amount of features that fall into each disjoint category, also called bins. Let b denote the total number of bins, FLE is a function p_k that must satisfy:

$$l+1 = \sum_{k=1}^{b} p_k, \ k = 1..b. \tag{2}$$

The number of bins can be variously predefined depending on application assumptions. The more bins, the finer detection. Anyway, b should be smaller than l to make sense of a histogram. If there exist outliers with values far from the mean, the frequency p_k at bin k will have the highest value. However, the position of the maximum frequency is highly dynamic and depends on which bin most samples fall into as shown in Figure 1. Along the horizontal axis, the most left bin (called least significant bin LSB) contains smallest values, and the most right bin (called most significant bin MSB) contains highest values. Along the vertical axis, the maximum frequency is one, and the minimum is zero. Looking at Figure 1, we also reckon that the distances from outliers to the means of distribution vary largely, such as from 0.3 to 11. As we discussed earlier, it is hard to use KL divergence to determine a distance threshold. Meanwhile, that the values of frequencies are almost similar in all examples (a), (b) and (c) makes it feasible to chose a constant threshold. This is an advantage of FLE over other approaches, which can overcome the issue caused by the dynamic of bin widths.

Since the mean jumps unpredictably, repeatedly finding the location of the means consequently overloads the mobile platforms. To overcome this issue, FLE takes the absolute values of given data as the input, for example $|\mathcal{X}|^t$ in our problem. As a result, this technique always pushes the mean and outliers to opposite sides along the horizontal axis. Therefore, we only need to simply count the frequencies of LSB and MSB. We estimate the anomaly probability of test data $|x|_i^t$ for measurement channel i, denoted by $\bar{p}(|x|_i^t)$, by the frequency sum of these two bins:

$$\bar{p}(|x|_i^t) = p_1 + p_l, \tag{3}$$

where p_1 is the frequency of LSB and p_l is the frequency of MSB.

Aggregating individual anomaly probability for all measurement channels depends on types of observing contexts. For instance, a bump in the road would be an interesting event, but it affects significantly the accelerometer and no other sensors. Choosing the max probability is a suitable option,

$$\bar{p}(|x|^t) = max\{\bar{p}(|x|_i^t)\}, \ i = 1..n. \tag{4}$$

ϵ	F_1 score
0.1	0.71605
0.2	0.87218
0.3	0.78378
0.4	0.37908
0.5	0.14428

Table 1: Tuning ϵ with F_1 score.

Conversely, if an event affects a set of sensor types, for example, a blast might be sensed by microphones, light, accelerometer, and pressure sensors, using sum or correlation among probabilities is a better choice.

To match Equation 4 with the problem definition Equation 1, we take the complement of the anomaly probability as the imaginary density probability:

$$p(x^t) \leftarrow 1 - \bar{p}(|x|^t). \tag{5}$$

Using above Equation 5, we can detect anomalies using the condition described by Equation 1.

Online FLEAD Algorithm

Algorithm 1 describes the pseudo code of our approach. At sequential time t, we have a dataset of measurements with n tuples $\delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$ from n channels. Each tuple has different dimension m_i because each measurement channel might have a different sampling rate. Absolute values of $l+1$ features are stored in a FIFO buffer $|\mathcal{X}|_i^t$, $i = 1..n$. Depending on specific applications, ϵ, m_i, l and b can be chosen differently. Firstly, for each tuple i (3), features are extracted from measurements (4). After that, we find the max and min of features (5) and count features that fall into LSB and MSB (6). The anomaly probability of test data x_i^t for measurement channel i then is computed by Equation 3 based on proposed FLE (7). After updating $|\mathcal{X}|_i^{t-1}$, this process (3-9) is repeated for all n measurement channels.

Next, we test the density probability of x^t using Equation 4 and Equation 5. If $p(x^t)$ is smaller than the chosen ϵ (11), we conclude that the current data pattern is likely anomalous (12). In other words, it seems that there is some new event happening at current granularity t.

Algorithm 1 <Online FLE Anomaly Detection>

1: INPUT: Dataset δ with n tuples with m_i measurements, l, b, ϵ, m_i, $i = 1, .., n$
2: OUTPUT: at sequential time t
3: **for** *each tuple* i **do**
4: extract feature $|x|_i^t \leftarrow \delta_i^k,\ k = 1, .., m_i$
5: compute max and min of $|\mathcal{X}|_i^t$
6: count features in LSB and MSB
7: compute $\bar{p}(|x|_i^t)$
8: update FIFO buffer $|\mathcal{X}|_i^{t-1} \leftarrow |x|_i^t$
9: **end for**
10: compute $p(x^t) \leftarrow 1 - \bar{p}(|x|^t)$
11: **if** $p(x^t) < \epsilon$ **then**
12: x^t is anomalous
13: **end if**<end>

Complexity

Given a dataset $\mathcal{X}^{t-1} = \{x^{t-1-l}, x^{t-l}, \ldots, x^{t-1}\}$ of l granularities and test data x^t, the total number of samples is $n \times (l+1)$. For each channel i, FLEAD uses $l+1$ features, and requires $l+1$ operations to find the min and max values. It also needs $l+1$ operations to count the anomaly probability, sum of samples falling into LSB and MSB. Therefore, the complexity of FLEAD is:

$$O(2(l+1)n) \simeq O(nl). \tag{6}$$

Since the complexity of other tools to detect anomalous data depends on their real applications, in this paper, we compute the complexity of a naive anomaly detection based on Multivariate Gaussian Distribution (MVN) with above given input. MVN first needs $n \times l$ operations to compute the mean vector $\mu = \frac{1}{l}\sum_{k=1}^{m} x^k$. MVN also needs l^2 operations to compute covariance matrix of a pair channels (i, j). Since there are n channels, MVN needs $n^2 l^2$ operations to compute the final covariance

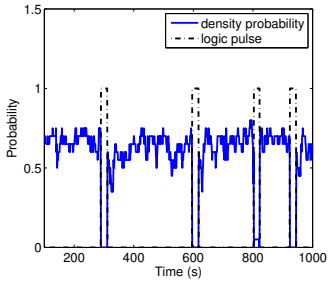

Figure 2: Density probability and logic pulse.

$\triangle$	l	precision	recall
0.5s	10	0.78667	1.00000
	20	0.85507	1.00000
	50	0.98333	0.98333
1.0s	10	0.85507	1.00000
	20	0.92188	1.00000
	50	0.95161	0.83099

Table 2: Edge detection for 1-minute anomaly intervals.

$\triangle$	l	precision	recall
0.5s	10	0.73333	1.00000
	20	0.73333	1.00000
	50	0.78571	1.00000
1.0s	10	0.78571	1.00000
	20	0.84615	1.00000
	50	0.57895	1.00000

Table 3: Edge detection for 5-minute anomaly intervals.

matrix $\Sigma = \frac{1}{l-1}\sum_{k=1}^{m}(x^k-\mu)(x^k-\mu)^T$. To calculate the density probability of test data x^t,

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{7}$$

inverting Σ requires n^3 operations. Therefore, the complexity of MVN in our problem is:

$$O(nl+n^2l^2+n^3) \simeq O(n^2l^2+n^3). \tag{8}$$

From Equation 6 and Equation 8 we conclude that FLEAD consumes much fewer computations than MVN does.

Empirical Study

We investigate FLEAD with experimental data that are collected with Samsung Galaxy Note II, Android $4.1.1$. Data were simultaneously collected from 19 channels of 7 sensors: acceleration (x, y and z), gyroscope (x, y and z), magnetic (x, y and z), barometer (pressure), gravity (x, y and z), linear acceleration (x, y and z), and orientation (x, y and z). To our experience, the most challenging anomaly detection is with smartphones movements. Onboard sensors in smartphones have low sampling rate, around 50 Hz when using SENSOR_DELAY_UI (60,000 microsecond delay) for the gyroscope. Other sensors except microphones have even lower sampling rate. However, smartphones usually move quite fast and unpredictably. Therefore, in this paper, we present the result of an experiment mainly related to movement sensors (acceleration, gyroscope, gravity, linear acceleration, and orientation).

For each period of one hour, we randomly shake the mobile phones at interval time around 1, 5 or 10 minutes to create anomalous data, and then laying them back on a table. As a result, three sets of one-hour data with different occurring frequencies of anomalies were obtained. This scenario is quite simple yet useful to investigate the algorithm. It produces similar results as other scenarios, including but not limited to, withdrawing a smartphone out of a pocket, sitting down and standing up, walking and running, falling from a bike. Note that we only want to detect the event, not its nature.

To extract features from measurements, we use the means of measurements that fall outside of the confidence interval (significant level is set 5% in our experiment). Since the variance of the unconfidence means is always greater than that of the conventional means, the feature is more sensitive to outliers.

Since the maximum sampling rates among 19 measurement channels is 50 samples per second, we only consider granularity (time window) of 0.5 and 1.0 second in our experiments. Granularity above 2 seconds would cause a considerable delay in most real-time applications. In addition, we choose the length of historical unconfidence means l as 50 maximum, because the anomaly interval can be less than one minute in our experiments. Let the number of bins b be 10 for all experiments so that l has space to vary from 10 to 50 ($l \geq b$).

To choose a threshold value for ϵ, we tune ϵ from 0.1 to 0.5 (half of maximum probability) with an arbitrary dataset. Using F_1 score in statistic, we obtain an optimal epsilon that can be used for all experiments in this paper. Table 1 shows F_1 values of FLEAD in case of 1-minute anomaly intervals when tuning ϵ. The table shows that 0.2 is an optimal value for ϵ, which generates the highest score, 0.87218.

$\triangle$	l	precision	recall
0.5s	10	0.62500	1.00000
	20	0.83333	1.00000
	50	1.00000	1.00000
1.0s	10	0.83333	1.00000
	20	0.83333	1.00000
	50	0.62500	1.00000

Table 4: Edge detection for 10-minute anomaly intervals.

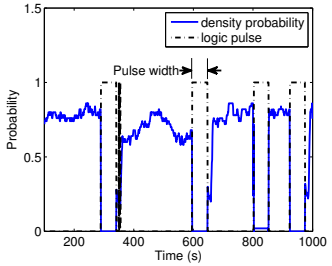

Figure 3: Pulse correction to improve anomaly detection.

$\triangle$	l	precision	recall
0.5s	10	0.93651	1.00000
	20	0.96721	1.00000
	50	0.98333	1.00000
1.0s	10	0.95161	1.00000
	20	1.00000	1.00000
	50	1.00000	0.83099

Table 5: Pulse width detection for 1-minute anomaly intervals.

To visualize the anomaly probability, we generate a logic pulse $(0-1)$, denoted by $\tilde{p}(x^t)$, according to

$$\tilde{p}(x^t) = \begin{cases} 0, & \text{if } p(x^t) \geq \epsilon \\ 1, & \text{if } p(x^t) < \epsilon \end{cases}. \quad (9)$$

Figure 2 demonstrates a portion of the logic pulse $\tilde{p}(x^t)$ generated from an estimate density probability $p(x^t)$ with the chosen threshold $\epsilon = 0.2$. Based on the logic pulse, we count the number of pulses (true positive, false positive, and false negative) to evaluate FLEAD's performance in terms of precision and recall. The counting method relies on either edge detection (raising and falling edges) or pulse width detection.

For edge detection based counting, Table 2, 3 and 4 list precision and recall values on three one-hour datasets. In general, the results show that FLEAD is significantly sensitive to anomalies as we expected as the recall values are very close to 1. That a recall value in Table 2 is a little below 1 is due to the exceeded length l. With $l = 50$ and $\triangle = 1$ s, we have the length of historical data $l \times \triangle = 50$ s. The anomaly intervals are around 60 s. Therefore, it is possible that the length of historical data sometimes exceeds the time betweens two consecutive anomalies. In such case, FLEAD considers the latter anomaly as normal data. When the anomaly intervals increase, such as 5 and 10 minutes in Table 3 and 4 respectively, this side effect totally vanishes and FLEAD gives perfect recall values.

Table 2, 3 and 4 also show that FLEAD points out anomalies as expected, especially when increasing l or $\triangle$. The more measurement data, the better estimation. However, if l and $\triangle$ increase too much together, for example, $l = 50$ and $\triangle = 1$ s in our experiments, it might generate glitches of density probability right after the anomalies data is removed from the FIFO buffer $|\mathcal{X}|_i^t$. This can lead to unexpected logic pulses right after the correct ones as shown in Figure 3. Using the second technique for counting based on the pulse width will solve this issue because these fake glitches are ignored.

We also observed that precision in case of long anomaly intervals can be lower than that in case of shorter anomaly intervals. This can be explained. The precision actually is not effected by the period of anomalies as long as the period is longer than l. The lower precision is due to fewer number of actual anomalies in latter tables. In particular, Table 2, 3 and 4 results are respectively obtained by experiments with 59, 11 and 5 actual anomalies per hour. Therefore, a false positive, failing to detect that there is no anomaly, will affect the precision more if there are fewer true anomalies.

As discussed with edge detection, the pulse width based technique indicates better true anomalies. Since a pulse width is defined by the period of an anomaly existing in the FIFO buffer $|\mathcal{X}|_i^{t-1}$, the width of a pulse that represents the true anomaly is equal l. Adding this pulse width constraint, FLEAD performs much better as shown in Table 5, 6 and 7. Since the glitches are skipped, precision is significantly improved.

Finally, we also implemented a naive anomaly detection algorithm based on online Multivariate Gaussian Distribution (MVN). The results in Table 8, show that MVN totally fails to detect anomalies in our experimental data even with the best threshold tuned with F_1 score, $\epsilon = 0.5$, which FLEAD performs very well. The reason is that there were insufficient samples for training.

For other events, such as sounds or temperature, the algorithm detects anomalies even better. Although sounds

$\triangle$	l	precision	recall
0.5s	10	0.84615	1.00000
	20	0.91667	1.00000
	50	0.91667	1.00000
1.0s	10	0.91667	1.00000
	20	0.91667	1.00000
	50	0.91667	1.00000

Table 6: Pulse width detection for 5-minute anomaly intervals.

$\triangle$	l	precision	recall
0.5s	10	1.00000	1.00000
	20	1.00000	1.00000
	50	1.00000	1.00000
1.0s	10	1.00000	1.00000
	20	1.00000	1.00000
	50	1.00000	1.00000

Table 7: Pulse width detection for 10-minute anomaly intervals.

$\triangle$	dataset	tp	fp	fn
0.5s	$1min \backslash l = 50$	0	62	59
	$5min \backslash l = 250$	0	42	11
	$10min \backslash l = 550$	0	4	5
1.0s	$1min \backslash l = 50$	0	44	59
	$5min \backslash l = 250$	0	14	11
	$10min \backslash l = 550$	0	1	5

Table 8: True positive (tp), false positive (fp) and false negative (fn) of 1-minute, 5-minute and 10-minute at upper bound lengths of Multivariate Gaussian Distribution.

is dynamic, it is sampled with high sampling rate, at least 8 kHz. Temperature is mostly stable and changed slowly. Indeed, we also tested with sounds such as dog barking, car horning, baby crying, etc., with several noise backgrounds and the algorithm works quite well.

Conclusion

Continuous sensing applications with mobile platforms encounter many resource constraints. Using anomaly detection to trigger power consuming processes is a good way to save battery and computation. However, most existing anomaly detection algorithms are power hungry and memory consuming since they are developed for processing big data. Moreover, lack of training samples and continuous connectivity due to device's mobility prevents using such conventional approaches. This paper presents an energy efficient algorithm (FLEAD), which is online and featherweight.

Data split into small granularity time windows allows online data processing with mobile platforms. By capturing the likelihood of the traditional frequency histogram, FLEAD efficiently detects anomalies with high precision and sensitivity. The results from our empirical study on mobile platforms (Android phones) show that FLEAD is able to detect anomalous patterns in data. Note that FLEAD is also suitable for large-scale heterogeneous sensor networks, from mobile devices to fixed infrastructures. We also plan to further investigate FLEAD with more existing algorithms and complicated events, such as a blast which changes pressure, temperature, vibration, and sounds all at the same time.

Acknowledgements

This work is supported by the SenSafety project in the Dutch Commit program, www.sensafety.nl.

References

[1] Das, T., Mohan, P., Padmanabhan, V. N., Ramjee, R., and Sharma, A. Prism: Platform for remote sensing using smartphones. In *Proc. MobiSys*, ACM (2010), 63–76.

[2] Lane, N. D., Xu, Y., Lu, H., Hu, S., Choudhury, T., Campbell, A. T., and Zhao, F. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proc. UbiComp*, ACM (2011), 355–364.

[3] Le, V.-D., Scholten, H., and Havinga, P. Unified routing for data dissemination in smart city networks. In *Proc. IoT* (2012), 175–182.

[4] Eskin, E. Modeling system calls for intrusion detection with dynamic window sizes. In *Proc. DISCEX* (2001).

[5] Fawcett, T., and Provost, F. Activity monitoring: Noticing interesting changes in behavior. In *Proc. SIGKDD* (1999), 53–62.

[6] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. 41*, 3 (2009), 15:1–15:58.

[7] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection for discrete sequences: A survey. *Knowledge and Data Engineering, IEEE Transactions on 24*, 5 (2012), 823–839.

[8] Manevitz, L. M., and Yousef, M. One-class svms for document classification. *J. Mach. Learn. Res. 2* (2002), 139–154.

[9] Srivastava, A. Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques. In *Proc. Aerospace Conference* (2006).

[10] Ng, A. Machine learning. https://class.coursera.org/ml/lecture/97\/.