An Adaptive Abnormal Behavior Detection using Online Sequential Learning

Rei Ito*, Mineto Tsukada*, Masaaki Kondo†, Hiroki Matsutani*
*Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan 223-8522
Email: {rei,tsukada,matutani}@arc.ics.keio.ac.jp
†The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan 113-8656
Email: kondo@hal.ipc.i.u-tokyo.ac.jp

***Abstract*—In the real world, normal and abnormal behavior patterns vary depending on a given environment, which means that the abnormal behavior detection model should be customized. To address this issue, in this paper, we employ OS-ELM (Online Sequential Extreme Learning Machine) and Autoencoder for adaptive abnormal behavior detection. First, state-transition probability tables of a target during an initial learning period are learned as normal behaviors. Then, Autoencoder-based anomaly detection is performed for the state-transition probability tables of subsequent time frames. The abnormal behavior detection model is updated by using OS-ELM algorithm every time a new probability table or behavior comes. The number of abnormal behavior detection instances is dynamically tuned to reflect the recent normal patterns or modes. Also, the table is compressed to reduce the computation cost. Evaluation results using a driving dataset of cars show that the proposed abnormal behavior detection accurately identifies normal and aggressive driving patterns with the optimal number of the abnormal behavior detection instances.**

***Index Terms*—Machine learning, Abnormal behavior detection, OS-ELM, Autoencoder, and Clustering**

I. Introduction

Abnormal behavior detection is used to detect unusual behavior or state transition of targets. It can be applied to detect infected or compromised networked systems [1], aggressive driving of cars [2], and various surveillance applications [3]. In the real world, normal and abnormal behavior patterns vary depending on a given environment. A normal behavior in an environment may be abnormal in different environments or situations. Thus, the abnormal behavior detection model should be customized to a given environment. To address this issue, in this paper, we employ OS-ELM (Online Sequential Extreme Learning Machine) [4] as an online sequential learning algorithm to adaptively update the model. It is also combined with Autoencoder [5] for the adaptive unsupervised abnormal behavior detection.

In the proposed approach, first, state-transition probability tables of a target during an initial learning period are learned as normal behaviors. Then, Autoencoder-based anomaly detection is performed for the state-transition probability tables of subsequent time frames, in order to detect abnormal patterns. The abnormal behavior detection model is updated by using OS-ELM algorithm every time a new probability table or behavior comes.

In practice, abnormal behaviors should be accurately detected from multiple normal patterns. To improve the accuracy of abnormal behavior detection in such cases, we introduce multiple adaptive abnormal behavior detection instances, each of which is specialized for each normal pattern. Since these abnormal behavior detection instances should be dynamically tuned, we propose a reconfiguration technique of these instances, in order to adapt to the recent normal patterns or modes. Also, the input state-transition probability tables are typically sparse, so we propose a probability table compression technique, in order to reduce the computation cost and memory requirement. Evaluation results using a driving dataset of cars show that the proposed abnormal behavior detection accurately identifies normal and aggressive driving patterns with the optimal number of the abnormal behavior detection instances.

The rest of this paper is organized as follows. Section II surveys existing work on abnormal behavior detection and introduces OS-ELM algorithm. Section III proposes the abnormal behavior detection using multiple OS-ELM instances, reconfiguration of the instances, and compression of the state-transition probability table. Section IV evaluates the proposed approach using two abnormal behavior detection applications. Section V concludes this paper.

II. Baseline

In this section, first, a conventional abnormal behavior detection approach is introduced. Then, baseline technologies of the proposed approach are introduced.

A. Abnormal Behavior Detection using Hidden Markov Model

In this paper, state-transition probabilities of a target are monitored and used for the abnormal behavior detection. In this case, Hidden Markov Model (HMM) [6] is useful to represent the probabilistic model of the behavior assuming a Markov process with unobservable states. That is, state-transition probabilities of normal behaviors are learned first. Then, state-transition probabilities of subsequent behaviors are evaluated in order to detect unusual behaviors.

To learn normal behaviors, the HMM parameters are extracted from normal state-transition probabilities. A conventional approach for the HMM parameter estimation is Baum-Welch algorithm [7], which is known as an EM (Expectation-Maximization) algorithm. EM algorithm is iterative. That is, it estimates the parameters of the probabilistic model by maximum likelihood (MAP) estimation by repeatedly alternating two steps: E-step for calculating an expected value of the likelihood of the model and M-step for calculating model parameters that maximize the expected value. These two steps are continuously performed until convergence. In practice, a mixture of HMMs is used to improve recognition performance as in [8]. In their approach, an online parameter estimation of HMMs is performed to dynamically generate the description of behaviors.

In this paper, as an alternative approach for the conventional HMM-based abnormal behavior detection, we propose a neural-network based online algorithm for the abnormal behavior detection that can update the model so as to adapt to a given environment. More specifically, a combination of OS-ELM and Autoencoder [9] is used for such adaptive unsupervised abnormal behavior detection. In the following subsections, OS-ELM and Autoencoder are introduced briefly as baseline technologies of the proposed approach.

B. *Online Sequential Extreme Learning Machine (OS-ELM)*

OS-ELM [4] is an online sequential learning algorithm for single-layer feedforward networks that consist of input layer, hidden layer, and output layer. The numbers of their nodes are denoted as n, m, and n', respectively. It sequentially learns input data $x \in \boldsymbol{R^{k \times n}}$, where k denotes a batch size. As in [9], k is fixed at one in this paper in order to eliminate the pseudo inverse operations of $k \times k$, except for initial training phase mentioned later. This is beneficial since the pseudo inverse operation is the major bottleneck of this approach. Please note that, in the abnormal behavior detection, x is a state-transition probability table of a target in a single time frame.

For the i-th input data $x_i \in \boldsymbol{R^{1 \times n}}$, the hidden layer matrix is defined as $h_i \equiv G(x_i \cdot \alpha + b)$ using activation function G, weight matrix $\alpha \in \boldsymbol{R^{n \times m}}$ between the input and hidden layers, and bias $b \in \boldsymbol{R^m}$ of the hidden layer. Here, α is initialized with random values. Then, the optimized weight matrix $\beta_i \in \boldsymbol{R^{m \times n'}}$ between the hidden and output layers can be computed by the following formulas using intermediate result P_i and training data $t_i \in \boldsymbol{R^{n'}}$.

$$\begin{aligned} \boldsymbol{P_i} &= \boldsymbol{P_{i-1}} - \frac{\boldsymbol{P_{i-1} h_i^T h_i P_{i-1}}}{1 + \boldsymbol{h_i P_{i-1} h_i^T}} \\ \boldsymbol{\beta_i} &= \boldsymbol{\beta_{i-1}} + \boldsymbol{P_i h_i^T} (\boldsymbol{t_i} - \boldsymbol{h_i \beta_{i-1}}) \end{aligned} \tag{1}$$

In particular, the initial values P_0 and β_0 are obtained as follows.

$$\begin{aligned} \boldsymbol{P_0} &= (\boldsymbol{H_0 H_0^T})^{-1} \\ \boldsymbol{\beta_0} &= \boldsymbol{P_0 H_0^T t_0} \end{aligned} \tag{2}$$

Here, H_0 is the initial hidden layer matrix obtained from the initial data $x_0 \in \boldsymbol{R^{k_0 \times n}}$, where k_0 is the initial batch size which should be greater than m.

In OS-ELM algorithm, weights β_i and their intermediate results P_i are computed from previous learning results β_{i-1} and P_{i-1}. Thus, it can sequentially update the model every time a new state-transition probability table of a target is fed.

C. *Autoencoder*

Autoencoder [5] is a type of neural network used for dimensionality reduction, learning of generative models, and unsupervised anomaly detection.

In this paper, Autoencoder is combined with OS-ELM to perform the adaptive unsupervised abnormal behavior detection. In this case, n and n' are the same. Input data x_i is used also as training data t_i. That is, the weight β_i is trained so that input data x_i is reconstructed by Autoencoder. Assume the Autoencoder has been trained only with normal state-transition probabilities. In this case, the difference between the input data and reconstructed data (denoted as loss value) should be small when the input data is close to the normal patterns. On the other hand, the loss values become large when unusual probabilities are fed; thus the abnormal behaviors can be detected.

III. Abnormal Behavior Detection using Online Sequential Learning

In this paper, a combination of OS-ELM and Autoencoder is used for the adaptive unsupervised abnormal behavior detection that can adaptively update the model depending on a given environment. A state-transition probability table of a target in a single time frame is used as an input for the abnormal behavior detection.

As in a mixture of multiple HMMs used for the improved abnormal behavior detection, the proposed approach introduces multiple pairs of OS-ELM and Autoencoder, each of which is denoted as an "instance". Assuming k is the number of the instances, the multi-instance version can accurately identify at least k normal patterns. It can improve the detection accuracy especially when the target has multiple normal modes. In the adaptive abnormal behavior detection, k should be dynamically tuned so as to follow the number of normal modes of the target. Also, the state-transition probability table should be compressed to reduce the computation cost and memory requirement.

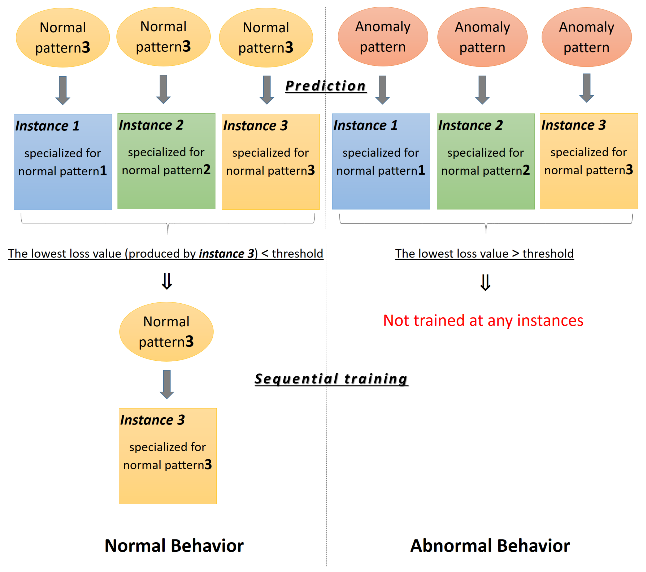
In the following subsections, 1) the multi-instance design of the OS-ELM and Autoencoder based abnormal behavior detection, 2) the dynamic reconfiguration technique of the instances based on the optimal k, and 3) the compression technique of the state-transition probability table are proposed.

A. *Multi-Instance Design*

In real environments, normal patterns are not always simple, necessitating a mixture of multiple models for the abnormal behavior detection. The basic idea is that, assuming multiple normal patterns, a single OS-ELM and Autoencoder instance is assigned to each normal pattern. In other words, each instance is specialized for each normal pattern in order to clearly distinguish anomaly patterns from multiple normal patterns.

In the multi-instance version, a prediction is performed at all the instances, while the sequential training is done at a single instance which is in charge of the input pattern. Thus, a sequential training is completed with the following two steps.

1) A prediction is performed at all the k instances. The lowest loss value among them is the prediction result. When the lowest loss value exceeds a given threshold, the input data (i.e., state-transition probability table) is detected as anomaly; otherwise detected as normal.



Normal pattern3
Normal pattern3
Normal pattern3
Anomaly pattern
Anomaly pattern
Anomaly pattern
Prediction
Instance 1 specialized for normal pattern1
Instance 2 specialized for normal pattern2
Instance 3 specialized for normal pattern3
Instance 1 specialized for normal pattern1
Instance 2 specialized for normal pattern2
Instance 3 specialized for normal pattern3
The lowest loss value (produced by instance 3) < threshold
The lowest loss value > threshold
Normal pattern3
Not trained at any instances
Sequential training
Instance 3 specialized for normal pattern3
Normal Behavior
Abnormal Behavior

Fig. 1. Multi-instance design of adaptive abnormal behavior detection.

2) When the prediction result is normal, a sequential training is performed at the instance that produces the lowest loss value.

Please note that abnormal data is not trained at any instances unless the instances and their coverage are reconfigured (see also Section III-B).

Assuming k is given by users, coverage of the k instances should be properly configured in order to clearly distinguish anomaly patterns from multiple normal patterns. Below is the initialization procedure for the k instances.

1) A certain amount of input data is accumulated. The amount depends on a buffer size.
2) A clustering is performed for the accumulated input data so that the accumulated data is broadly partitioned into k clusters.
3) Each cluster is assigned to each instance. Then each instance is trained with its assigned input data.

The simplest approach to broadly partition the accumulated data is K-means algorithm when k is given. When k is not specified but density parameters of clusters are given, DBSCAN algorithm [10] or, for high-dimensional data, SUBCLU algorithm [11] can be used.

B. Reconfiguration of Multiple Instances

The instance configuration procedure mentioned above is intended to be performed at the initialization step. In addition, a runtime reconfiguration of the instances is useful when a cluster of new patterns appears frequently and finally becomes a new normal cluster. In fact, there is no guarantee that all the normal patterns are accumulated during the initialization phase. Also, an existing normal cluster may disappear as time goes by.

To address these issues, a dynamic reconfiguration of multiple instances based on an estimated k parameter is required. Below is the procedure.

1) A certain amount of input data is accumulated. The amount depends on a buffer size.
2) Three clustering trials are performed for the accumulated input data so that the accumulated data is partitioned into $k-1$, k, and $k+1$ clusters, respectively.
3) A new k parameter is selected based on a quantitative evaluation of the three clustering trials.
4) Based on the new k parameter and its clustering result, each cluster is assigned to each instance. Then each instance is trained with its assigned input data.

This procedure is executed periodically or when the abnormal patterns appear frequently.

In step (3), Sum of Squared Error (SSE) and PseudoF [12] values are used to quantitatively evaluate the three clustering trials. SSE for k clusters is calculated as follows.

$$SSE_k = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x), \tag{3}$$

where x is a data point in cluster C_i, m_i is a centroid of cluster C_i, and $dist^2(m_i, x)$ is a squared distance between m_i and x. PseudoF is calculated as follows.

$$PseudoF = \frac{(SSE_1 - SSE_k)/(k-1)}{SSE_k/(n-k)}, \tag{4}$$

where n is the number of data points and SSE_1 is an SSE value when all the data points are in the same cluster.

PseudoF value describes the ratio of between-cluster variance to within-cluster variance. A larger PseudoF value means a better clustering. In step (3), a new k parameter is selected from $k-1$, k, and $k+1$ that maximizes the PseudoF value. Please note that PseudoF value cannot be calculated when $k < 2$. In this case, SSE value is used instead.

C. Compression of State-Transition Probability Table

In this paper, a series of behaviors of a target in a time frame are represented by a state-transition probability table showing a state-transition probabilities between arbitrary two states. The left side of Figure 2 shows an example of a state-transition probability table for nine states. Each row represents a source state, while each column represents a destination state. For example, a state-transition probability from state-3 to state-2 is 5. These values are normalized between 0 to 1 when they are fed to the neural network.

When the number of states is s, the state-transition probability table size is s^2. As s increases, the state-transition probability table size and the numbers of input/output layer nodes of a neural network increase significantly, resulting in a high computation cost for the sequential training[1] and prediction. Also, the probability table is typically sparse because of some state-transitions that never occur.

The left side of Figure 2 shows a case where state-transitions within the same state or their neighboring states tend to be high compared to the other state transitions. That is, state-i is changed to state-$(i-1)$, state-i, or state-$(i+1)$. In this case, the table forms a band matrix that has nonzero elements

[1]The computation cost increases as the numbers of input/output layer nodes increase as modeled in [9].

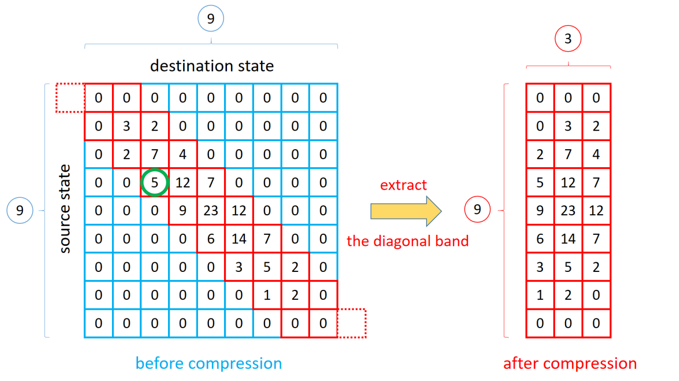


Fig. 2. Compression of state-transition probability table.

on the main diagonal (state-i to state-i), the first diagonal below this (state-i to state-$(i-1)$), and the first diagonal above this (state-i to state-$(i+1)$). We can obtain the compressed form of this state-transition probability table by extracting the diagonal band of the tridiagonal matrix. The result is illustrated in the right side of Figure 2. In Section IV-B, this compression technique is applied to a driving dataset of cars, where a state represents their speed in 15 levels (1 level = 10km/h).

IV. Experimental Results

In this section, the reconfiguration technique of multiple instances when multiple normal patterns exist is evaluated. Then the compression technique of the state-transition probability table is compared with the uncompressed design.

A. *Reconfiguration of Multiple Instances*

The multi-instance design and its dynamic reconfiguration technique are evaluated with the Schonlau dataset [13] that contains UNIX command histories of 50 users. Below is the experiment scenario of the multi-instance design and its dynamic reconfiguration technique using this dataset.

1) An input command history is generated by mixing the command histories of users 21 and 32. A series of state-transition probability tables is generated from this mixed command history. The generated data is denoted as "2-user input data".
2) The initial training step is completed with this 2-user input data by using two instances (i.e., $k = 2$).
3) Another input command history is generated by mixing the command histories of users 21, 32, 46, and 17. A series of state-transition probability tables is generated from this mixed command history. The generated data is denoted as "4-user input data".
4) The sequential training step using this 4-user input data is started with the original two instances.

The buffer size for the dynamic reconfiguration is set to 280. We are thus expecting that the number of instances k is adjusted at every 280 input data and finally it is converged with four, since the input data is based on four users' command histories. Since the length of command histories in the dataset is not enough, we augmented the 2-user and 4-user input data, based on their original state-transition probabilities.

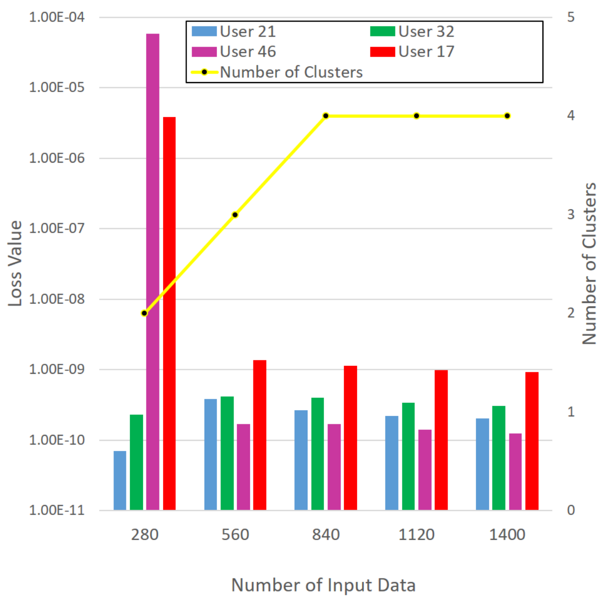


Fig. 3. Evaluation result of dynamic reconfiguration of multiple instances.

Here, the size of state-transition probability table is set to 1,024 assuming 32 types of UNIX commands. Thus, the numbers of input/output layer nodes of the neural network are both 1,024. The number of hidden layer nodes is set to 256.

Figure 3 shows the evaluation result of the experiment scenario. X-axis represents the number of input data. The instance reconfiguration is done at every 280 input data. A yellow line (Y-axis on the right side) represents the number of instances k. As shown, k is increased from two and it is finally converged with four, which means that the proposed instance reconfiguration technique accurately reflects the given normal patterns or modes.

In addition, Y-axis on the left side of Figure 3 shows the loss values for the four users. When $k = 2$, the loss values of users 46 and 17 are quite high, which means that these two users are not recognized as normal users. As the number of instances k increases, their loss values are decreased and finally converged when $k = 4$, which means that these four users are recognized as normal users.

B. *Compression of State-Transition Probability Table*

The compression technique of the state-transition probability table is evaluated with the UAH-DriveSet [14] that contains car driving histories of six drivers simulating three different behaviors: aggressive, drowsy, and normal. It is used for the aggressive driving detection. Their car speed is extracted from the GPS data obtained from a smartphone fixed in their cars. The sampling frequency of the car speed is 1Hz.

Below is the experiment scenario of the compression technique of the state-transition probability table using this dataset. Here, the aggressive, drowsy, and normal driving patterns are denoted as A, D, and N patterns, respectively.

1) Using the uncompressed state-transition probability tables, the initial training step is completed with A and D patterns (denoted as "AD" pattern). Then the sequential

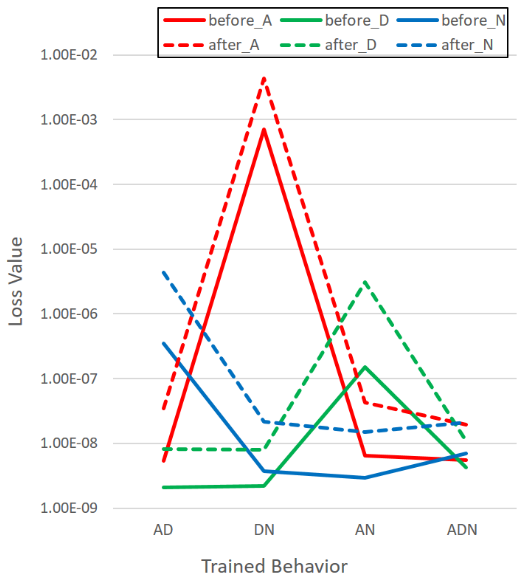


Fig. 4. Evaluation result of compression technique of probability table.

training step is started with A, D, and N patterns. Their loss values are denoted as before_A, before_D, and before_N, respectively. In addition to "AD" pattern, the same procedure is performed for "DN", "AN", and "ADN" patterns.

2) Using the compressed state-transition probability tables, the same procedure with (1) is performed. Their loss values are denoted as after_A, after_D, and after_N.

Then the loss values before and after the compression are compared (e.g., before_A vs. after_A) to see the accuracy degradation by the table compression.

Here, the size of the original state-transition probability table is set to 225 assuming 15 levels of car speeds (1 level = 10km/h). Thus, the numbers of input/output layer nodes of the neural network are both 225. The number of hidden layer nodes is set to 16. The original probability table is compressed with the same technique as shown in Figure 2. That is, the probability table is approximated as a tridiagonal matrix, and only the diagonal band is extracted, resulting in a 15×3 matrix. In this case, the numbers of input/output layer nodes are only 45. The number of hidden layer nodes is set to 16.

Figure 4 shows the result. X-axis represents the training patterns including "AD", "DN", "AN", and "ADN" patterns. Y-axis represents the loss values of before_A, before_D, before_N, after_A, after_D, and after_N. As shown, only the loss values of before_A and after_A for "DN" pattern are quite high, which means that the aggressive driving patterns are detected. The important thing is that the same tendency is observed before and after the compression in all the cases. This means that the proposed compression technique significantly reduces the numbers of input/hidden/output layer nodes of the neural network, while keeping the accuracy of the abnormal behavior detection.

V. Conclusions

In the real world, normal and abnormal behavior patterns vary depending on a given environment, and therefore the abnormal behavior detection model should be customized to the environment. In this paper, a combination of OS-ELM and Autoencoder was introduced for the adaptive unsupervised abnormal behavior detection that can adaptively update the model. In addition, a multi-instance design of the OS-ELM and Autoencoder pair was introduced for improving the detection accuracy. Then, a dynamic reconfiguration technique of these instances based on recent normal patterns or modes was proposed. A compression technique of the state-transition probability table was also proposed.

The evaluation results using the UNIX command histories of four users demonstrated that the proposed instance reconfiguration technique accurately reflects the recent normal patterns. Also, the evaluation results using aggressive, drowsy, and normal car driving patterns demonstrated that the proposed compression technique significantly reduces the numbers of input/hidden/output layer nodes of the neural network, while keeping the accuracy of the abnormal behavior detection.

Acknowledgements This work was supported by JST CREST Grant Number JPMJCR1785, Japan.

References

[1] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks," *Expert Systems with Applications*, vol. 29, no. 4, pp. 713–722, Nov. 2005.

[2] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? Presenting the public UAH-DriveSet," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC'16)*, 2016, pp. 387–392.

[3] T.-H. Yu and Y.-S. Moon, "Unsupervised Abnormal Behavior Detection for Real-Time Surveillance using Observed History," in *Proceedings of the IAPR Conference on Machine Vision Applications (MVA'09)*, 2009, pp. 9–16.

[4] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[5] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul 2006.

[6] S. S. Joshi and V. V. Phoha, "Investigating Hidden Markov Models Capabilities in Anomaly Detection," in *Proceedings of the 43rd Annual Southeast Regional Conference (ACMSE'05)*, 2005, pp. 98–103.

[7] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Jul. 2011.

[8] J. Yin and Y. Meng, "Abnormal Behavior Recognition Using Self-Adaptive Hidden Markov Models," in *Proceedings of the 6th International Conference on Image Analysis and Recognition (ICIAR'09)*, 2009, pp. 337–346.

[9] M. Tsukada, M. Kondo, and H. Matsutani, "OS-ELM-FPGA: An FPGA-Based Online Sequential Unsupervised Anomaly Detector," in *Proceedings of the International European Conference on Parallel and Distributed Computing (Euro-Par'18) Workshops*, 2018, pp. 518–529.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'96)*, 1996, pp. 226–231.

[11] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *Proceedings of the International Conference on Management of Data (SIGMOD'98)*, 1998, pp. 94–105.

[12] T. Calinski and J. Harabasz, "A Dendrite Method for Cluster Analysis," *Communications in Statistics*, vol. 3, pp. 1–27, 1974.

[13] "Masquerading User Data," http://www.schonlau.net/intrusion.html.

[14] "The UAH-DriveSet," http://www.robesafe.com/personal/eduardo.romera/uah-driveset/.