IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 2, 2021, accepted March 13, 2021, date of publication March 23, 2021, date of current version March 31, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068172

Intelligent Anomaly Detection for Large Network Traffic With Optimized Deep Clustering (ODC) Algorithm

ANNIE GILDA ROSELIN[1,2], PRIYADARSI NANDA[1], SURYA NEPAL[2], AND XIANGJIAN HE[1], (Senior Member, IEEE)

[1]Department of Electrical and Data Engineering, University of Technology Sydney (UTS), Ultimo, NSW 2007, Australia
[2]Commonwealth Scientific and Industrial Research Organisation (CSIRO/Data61), Marsfield, NSW 2122, Australia

Corresponding author: Annie Gilda Roselin (Annie.ArockiaBaskaran@uts.edu.au; gilda77_83@yahoo.co.in)

This work was supported by the Data61 through the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Marsfield, Australia.

ABSTRACT The availability of an enormous amount of unlabeled datasets drives the anomaly detection research towards unsupervised machine learning algorithms. Deep clustering algorithms for anomaly detection gain significant research attention in this era. We propose an intelligent anomaly detection for extensive network traffic analysis with an Optimized Deep Clustering (ODC) algorithm. Firstly, ODC does the optimization of the deep AutoEncoder algorithm by tuning the hyperparameters. Thereby we can achieve a reduced reconstruction error rate from the deep AutoEncoder. Secondly, ODC feeds the optimized deep AutoEncoder's latent view to the BIRCH clustering algorithm to detect the known and unknown malicious network traffic without human intervention. Unlike other deep clustering algorithms, ODC does not require to specify the number of clusters needed to analyze the network traffic dataset. We experiment ODC algorithm with the CoAP off-path dataset obtained from our testbed and the MNIST dataset to compare our algorithm's accuracy with state-of-art clustering algorithms. The evaluation results show ODC deep clustering method outperforms the existing deep clustering methods for anomaly detection.

INDEX TERMS Deep learning, AutoEncoders, latent space view, anomaly detection, regularization, BIRCH clustering.

I. INTRODUCTION

Network traffic increase is directly proportional to increasing malicious activities on the internet. IoT plays a vital role in producing a massive number of network traffic datasets and creates significant challenges for detecting anomalies.

Anomaly detection in network traffic with machine learning is a rapidly growing research area [1]–[7]. Deep clustering techniques for anomaly detection use variations of AutoEncoder's latent representation with a k-means clustering algorithm. For example, Deep Embedding Clustering (DEC) [8], Improved Deep Embedding Clustering (IDEC) [9] and Deep Density-based Clustering (DDC) [10] use dense deep AutoEncoder, Deep Convolutional Embedded Clustering (DCEC) [11] and Deep Density-based Clustering-Data Augmentation (DDC-DA) [10] use convolutional AutoEncoder with k-means clustering, Gaussian mixture variational AutoEncoder (GMVAE) [12] practices variational AutoEncoder with k-means clustering. Most of these deep clustering techniques use the k-means clustering algorithm for the data clustering part, which in turn demands the number of clusters manually. In a real-time situation, predicting the number of clusters at the initial time (training the model) for a new dataset might not help discover new and unknown anomalies. To overcome this major limitation of the existing works, we use BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) in our ODC deep clustering technique. BIRCH has the advantage of intelligent cluster assignment and anomaly detection without human intervention. Also, a deep AutoEncoder reduces the dimensionality of the dataset irrespective of it has linear/non-linear data. The BIRCH clustering method is not getting much attention among the researchers on deep clustering methods. However, BIRCH

The associate editor coordinating the review of this manuscript and approving it for publication was Amir Masoud Rahmani.

has the capability of doing intelligent clustering on a vast dataset [13].

Our contributions are summarized as follows:

- Optimization of the deep AutoEncoder by tuning the hyper-parameters to achieve a reduced reconstruction error rate.
- We inferred a novel unsupervised anomaly detection algorithm ''ODC'' by incorporating the BIRCH clustering algorithm with the Latent representation of the enhanced deep AutoEncoder.
- Unlike other deep clustering algorithms, ODC does not require to specify the number of clusters needed to analyze the network traffic.
- ODC handles anomalies, including known and unknown attacks intelligently, for a huge dataset.
- We analyzed how the Branching factor value and the Threshold value of BIRCH influence the clustering accuracy and normalized mutual information score values.

We observed that our ODC clustering algorithm outperforms the existing deep clustering methods for anomaly detection. Moreover, ODC suits well for vast network traffic datasets where multiple scans of the datasets are not advisable since ODC has the BIRCH clustering algorithm's embedment. ODC incorporates the advantages of the BIRCH clustering algorithm. We achieved great clustering accuracy and normalized mutual index score for the anomaly detection process due to the combination of a deep AutoEncoder and the BIRCH clustering algorithm. Also, ODC put a stop to the need of domain experts to manually label the large datasets and explicitly specify the number of clusters needed for the dataset. Our proposed method differs from the state of the arts [14]–[17] and [18] in which we associated BIRCH clustering with our enhanced deep AutoEncoder. To preserve the data point's local structure, the StructAE [19] learns representations for each data point by minimizing reconstruction error with respect to itself. However, ODC achieves low reconstruction error rate by tuning the hyperparameters such as activation function and the regularization function. Hence, we prove that ODC preserves the data points' structure, leading to an intelligent clustering method to detect anomalies.

The rest of the paper is organized as follows. Section II provides the background information needed to understand the ODC clustering algorithm. The working principles of a deep AutoEncoder and the BIRCH clustering algorithm are explained in Section II-A and Section II-B, respectively. Section III describes the state of the art of deep clustering algorithms. The proposed deep clustering method is explained in Section IV. Section V describes the evaluation process of the proposed deep clustering method. Finally, we discuss the possible extension of our research in Section VI.

II. BACKGROUND

Anomaly Detection [20] is the strategy of recognizing uncommon occasions or perceptions which can raise doubts by being factually not the same as the remainder of the perceptions. Present-day organizations are starting to comprehend the significance of interconnected tasks to get their business's full image. Additionally, they have to react to quick-moving changes in information instantly, particularly if there should be an occurrence of cybersecurity dangers.

Unfortunately, there is no compelling method to deal with and break down, continually developing datasets physically. With the dynamic frameworks having various segments in a ceaseless movement where the ''normal'' conduct is continually reclassified, another proactive way to deal with distinguishing anomalous behavior is required [20].

Based on the dataset we use to train the machine learning model, anomaly detection varies in many real-world applications and academic research areas. With the emergence of sensor networks, processing data as it arrives has become a necessity [21]. Techniques have been proposed that can operate in an online fashion [22]; such techniques assign an anomaly score to a test instance as it arrives, but also incrementally update the model. Authors in [23] showcased the importance of anomaly detection in dynamic settings through a real-world application example, i.e., forest fire risk prediction. Also, they recommend redesigning the current models to be able to detect outlying patterns accurately and efficiently. More specifically, when there are many features, a set of anomalies emerge in only a subset of dimensions at a particular period. This set of anomalies may appear normal regarding a different subset of dimensions and periods.

Authors in [24] discussed the unavailability of financial data for fraud detection research and a methodology for synthetic data generation. They suggest that a universal technique in the domain of fraud detection is yet to be found due to the evolving change in the context of normality and labeled data unavailability. According to [25] much of the research is performed on simulated data (37 out of the 65 surveyed papers); in-vehicle network data and vehicular ad hoc network (VANET) data are seldom considered together to safeguard the connected vehicles (except for 1 out of the 65 surveyed papers); Connected vehicles safety research does not get the same amount of attention as cybersecurity research. It is observed that the anomaly detection domain has various promising research directions; many anomaly detection methods require a large amount of test data set for detecting anomalies [26]. The literature survey we conducted in anomaly detection motivates us to use the machine learning models to determine the abnormal behavior of the legitimate user in a private network.

Anomaly detection should be possible, utilizing the ideas of Machine Learning. It tends to be done in the following manners:

Supervised Anomaly Detection: This strategy requires a labeled data set with normal and abnormal examples for building a prescient model. The most well-known supervised methods incorporate supervised neural networks, support vector machine, k-nearest neighbors, Bayesian networks, and decision trees [27]. Supervised models are accepted to

give a more superior detection rate than unsupervised techniques because of their capacity to encode interdependencies between factors, alongside their capacity to join both earlier knowledge and information and to restore a certainty score with the model yield [2].

Unsupervised Anomaly Detection: This strategy does not require labeled training data. They assume that the vast majority of the system associations are normal traffic and just a modest quantity of rate is unusual and envision that noxious traffic is factually not quite the same as should be expected traffic [28]. In light of these two suspicions, groups of regular instances are thought to be ordinary, and rare data groups are sorted as an anomaly. The most popular unsupervised algorithms include K-means, AutoEncoders, GMMs (Gaussian Mixture Models), and PCAs (Principle Component Analysis) based analysis [29].

Deep learning is the subspace of machine learning that accomplishes great performance as they learn the detailed features of datasets with the help of neural networks [30]. The existing deep clustering techniques for anomaly detection merge a deep learning algorithm, and a clustering algorithm usually k-means clustering algorithms. With the observations of background studies and the research gap learned from related work in Section III, we proposed our ODC in SectionIV for intelligent anomaly detection.

A. DEEP AUTOENCODER

An AutoEncoder with more than one hidden layer is called a deep AutoEncoder. Deep AutoEncoders learn more complex features of the dataset since they have more layers than a simple AutoEncoder. The deep AutoEncoder intends to reconstruct the input with minimum reconstruction error. The encoding part, decoding part, and the latent representation part (compressed input) are the three essential parts of the deep AutoEncoder. The application of deep AutoEncoder is un-avoidable in network traffic analysis since it compresses a sizeable high dimensional dataset into a low dimensional dataset.

For a given training [31] dataset $X = \{x_1, x_2, \ldots, x_m\}$ with m samples, where x_i is a d-dimensional feature vector, the encoder maps the input vector x_i to a hidden representation vector h_i through a deterministic mapping f_θ as given in (1)

$$h_i = f_\theta(x_i) = \sigma(Wx_i + b) \tag{1}$$

where, W is a $d \times d$ matrix, d is the number of hidden units, b is a bias vector, θ is the mapping parameter set $\theta = \{W, b\}$. σ is a proper activation function. The decoder maps back the resulting hidden representation h_i to a reconstructed d-dimensional vector y_i in input space as

$$y_i = g_{\hat{\theta}}(x_i) = \sigma\left(\hat{W}h_i + \hat{b}\right) \tag{2}$$

where $\hat{W}$ is a $d \times \hat{d}$ matrix, $\hat{b}$ is a bias vector and $\hat{\theta} = \{\hat{W}, \hat{b}\}$ [31]. The goal of training the AutoEncoder is to minimize the difference between input and output. Therefore, a loss function is calculated by the following equation

$$L(x, y) = \frac{1}{m}\sum_{i=1}^{m} \|x_i - y_i\|^2 \tag{3}$$

where m is the total number of training dataset. The main objective is to find the optimal parameters ($h_i and \theta$) which can effectively minimize the difference between input and reconstructed output over the whole training set as

$$\theta = \{W, b\} = \arg_\theta \min L(x, y) \tag{4}$$

B. BIRCH CLUSTERING ALGORITHM

BIRCH, refers to Balanced Iterative Reducing and Clustering using Hierarchies, created in 1996 by Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH is best suited for large data sets or streaming due to the ability to find good clustering solutions with single scan data. Optionally, the algorithm can further scan through the data to improve clustering quality. BIRCH outperforms the existing clustering methods such as K-means and DBSCAN clustering algorithms [13] for handling large data sets.

According to [13], BIRCH is a multipath search tree, like the structure of a B+ tree. There are three kinds of nodes in a cluster-feature (CF) tree: Leaf, NonLeaf, and MinCluster. Three following parameters are engaged with the model. The first parameter is B (Branching factor), the greatest number of child nodes that a non-leaf node can hold. The second parameter is L, the most extreme number of child nodes that a leaf node suits. Furthermore, the third parameter is T (Threshold), the most extreme span estimation of the cluster. A CF tree is a set of three data points in a single cluster. These data points are as follows:

$$CF = (N, \overrightarrow{LS}, SS) \tag{5}$$

- Count (N): The number of information esteems in the cluster.
- Linear Sum ($\overrightarrow{LS}$): Aggregate the individual coordinates of the data points. This is a proportion of the area of the cluster.

$$\overrightarrow{LS} = \sum_{i=1}^{N} \vec{x}_i \tag{6}$$

- Squared Sum (SS): Aggregate the squared coordinates of the data points. This is a proportion of the spread of the cluster.

$$SS = \sum_{i=1}^{N} \left(\vec{X}_i\right)^2 \tag{7}$$

BIRCH has two phases:

- Phase 1: Building the CF tree. Load the network traffic data into the memory by building a cluster-feature (CF) tree. This phase will compress the initial CF tree only when this option is chosen at the training time.

- Phase 2: Global Clustering. Optional refinement of clusters which are obtained from phase 1 by applying an existing clustering algorithm on the leaves of the CF tree.

In view of the Additivity Hypothesis of CF [13], the CF estimation of the parent node is the aggregate of the CF estimations of its child nodes.

$$\mathbf{CF}_1 + \mathbf{CF}_2 = \left(N_1 + N_2, L\vec{S}_1 + L\vec{S}_2, SS_1 + SS_2\right) \quad (8)$$

III. RELATED WORK

In DEC [8], initial dense auto-encoder is prepared with limiting recreation mistake. At that point, as a clustering advancement arrange, the strategy repeats between processing a helper target conveyance from AutoEncoder depiction and limiting the Kullback-Leibler disparity to it. In IDEC [9], it is contended that the grouping loss of DEC undermines the component space; in this way, IDEC proposes the clustering loss and reproduction loss of the auto-encoder.

Deep clustering in [32] shows that a l_2 normalization on the latent representation of AutoEncoder makes the latent space more divisible and minimized in the Euclidean space. This significantly improves the clustering precision when k-means clustering is utilized on the latent representation. DDC [10] clustering technique reduces the dimension of the dataset with the help of deep convolutional AutoEncoder and t-SNE algorithm. Consequently, DDC applies density-based clustering on the result of the t-SNE (2-dimensional embedded data) algorithm without mentioning the number of clusters in advance. Deep clustering algorithms [33], [34] and DDC are using t-SNE for further dimensionality reduction of input data. The issue with t-SNE is that it does not safeguard the distances nor thickness between the data. Also, the compressed data cannot be assured to recreate the original input since there are no hyper-parameters to reduce the reconstruction error between the input data and the recreated data.

Recent works on convolutional AutoEncoder clustering such as [35]–[39] are most applicable for clustering image datasets, not for analysing network traffic datasets. DCEC [11] embraces a convolutional AutoEncoder and shows that it improves the clustering exactness of DEC and IDEC. Dealing the anomalies in credit card transactions [15] is done with the AutoEncoder and k-means clustering algorithm on the European bank transaction dataset. However, this work and the other works specified in this Section III has the problem of predicting the number of clusters after pre-training the AutoEncoder.

Our proposed algorithm ODC optimizes the pre-training process of deep AutoEncoder to reduce the reconstruction error. Furthermore, it uses BIRCH clustering to overcome the limitations of the existing deep clustering algorithms.

IV. PROPOSED DEEP CLUSTERING METHOD

ODC groups the network traffic data based on the Euclidean distance between the nodes so that we get more and more dynamic clusters as the network traffic passes on to the ODC model.

A. ENHANCED DEEP AUTOENCODER

The enhanced AutoEncoder model is constructed using the proper combination of activation function, regularizers, and optimization functions to reduce the reconstruction error value. Our enhanced AutoEncoder treats every input as self-reliant values, thereby reducing the over-fitting of training data. The ODC training phase requires unsupervised learning and fine-tuning the model parameters to enhance the efficiency of the model.

We used an ELU (Exponential Linear Unit) [40] activation function for all layers and Adamax optimization function for the enhanced AutoEncoder model.

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(\exp(x) - 1) & \text{if } x < 0 \end{cases} \quad (9)$$

ELUs have negative values that push the average of the functions closer to zero. Average functions close to zero allow faster learning as the gradient approaches the natural gradient. ELUs for negative net entries are saturated at a negative value. Besides, the likelihood of code interference for different concepts is less likely, as incomprehensible negative values of information avoid distributed codes. α is a hyper-parameter of ELU. Positively activated ELUs interact by activating the next layer of units. Thus the ELU activation function is well-suited for deep network models where vanishing gradient interferes with the learning of the model.

Dropout regularizer randomly dropping out nodes, thereby increasing the uniqueness of a node in the network. The co-adaptation of the features in the node is reduced by adopting a dropout regularizer in the network.

The number of hidden layers ($h_i = \{h_0, h_1, h_2, h_3, h_4\}$) in our enhanced AutoEncoder are five. Here, the latent space can be represented as,

$$h_2 = f_\theta(x_2) = \sigma(Wx_2 + b) \quad (10)$$

According to [41] the dropout function is defined as,

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \end{aligned} \quad (11)$$

In equation 11 $*$ signifies a component insightful product. For any layer l, $\mathbf{r}^{(l)}$ is a vector of self-governing Bernoulli irregular factors each of which has probability p of being 1. This vector is sampled and multiplied element-wise with the outputs of that layer, $\mathbf{y}^{(l)}$, to create the thinned outputs $\tilde{\mathbf{y}}^{(l)}$. These reduced features are then used as input to the next layer. This procedure is applied at each layer. If we apply dropout to the hidden layer with a probability value of p, the equation would be modified as follows (at training time):

$$\begin{aligned} \tilde{x} &\sim \text{Dropout}(x) \\ h &= f(W\tilde{x} + b) \\ \tilde{h} &\sim \text{Dropout}(h) \\ y &= g\left(W\tilde{h} + b\right) \end{aligned} \quad (12)$$

A loss function is calculated by the following equation

$$L(x, y) = \frac{1}{m} \sum_{i=1}^{m} \|x_i - y_i\|^2 + r \tag{13}$$

where m is the total number of training dataset. The column named "Train RE" in Table 1 refers to the reconstruction error rate during training time and Test RE means, reconstruction error rate during testing time of our enhanced deep AutoEncoder. The values in Table 1 shows, how our optimized deep AutoEncoder outperforms to reduce the reconstruction error rate both at training and testing time.

TABLE 1. Optimization of deep AutoEncoder.

Model	Train RE	Test RE
Baseline–linear, sgd	0.00178	0.00178
Baseline–relu, adamax	0.00062	0.00063
Baseline–relu, l1	0.00125	0.00125
Baseline–relu, l2	0.00084	0.00084
Baseline–relu, dropout	0.00063	0.00063
Baseline–ELU, dropout	**0.00061**	**0.00061**

B. OPTIMIZED DEEP CLUSTERING WITH BIRCH

The compressed representation of data points ($h_2 = f_\theta(x_2) = \sigma(Wx_2 + b)$) which are obtained from the enhanced deep AutoEncoder as explained in IV-A is feed into the BIRCH clustering algorithm. Each time the new data point is added to the CF tree by calculating the radius of the cluster. The radius of the cluster (R) is calculated as

$$R = \sqrt{\frac{\sum_{i=1}^{N} \left(\vec{X}_i - \vec{C}\right)^2}{N}} = \sqrt{\frac{N \cdot \vec{C}^2 + SS - 2 \cdot \vec{C} \cdot \overrightarrow{LS}}{N}}$$

$$R = \sqrt{\frac{SS}{N} - \left(\frac{\overrightarrow{LS}}{N}\right)^2}$$

The calculated R-value decides where to push the new data point. If $R < T$, then a new data point is pushed to the same leaf node. If $R > T$, then the new data point is formed as a new leaf node. Thereby the CF tree is built for all the data points in our training and testing data. If we divide the sum of data points by the number of data points, we can get the centroid of the cluster. The centroid ($\vec{C}$) of the cluster is calculated as

$$\vec{C} = \frac{\sum_{i=1}^{N} \vec{X}_i}{N} = \frac{\overrightarrow{LS}}{N}$$

Thereby we can calculate the distance between two clusters CF_i and CF_j

C. ODC OUTLIER HANDLING

We can set aside a fixed measure of disk/memory space for taking care of anomalies. Anomalies are leaf nodes of low thickness that are made a decision to be irrelevant concerning the general clustering design. At the point when we revamp the CF-tree by reinserting the old leaf nodes, the size of the new CF-tree is diminished in two different ways [13]. To begin with, we increment the limit esteem (Threshold T), subsequently permitting each leaf node to assimilate more focuses. Second, we treat some leaf nodes as potential anomalies and work them out to disk. An old leaf node is viewed as a potential anomaly in the event that it has far less data points than normal. An increment in the T value or a modification in the distribution considering the new data could well infer that the potential anomaly never again qualifies as an anomaly data point.

The data point whose Euclidean distance to the closest seed is larger than twice the radius of that cluster is treated as an anomaly [13]. As a result, the potential anomalies are examined to check on the off chance that they can be re-invested in the tree without making the tree develop in size. In Algorithm 1 steps from 1 to 4 explain how the compressed form of the input dataset has been made with the help of optimized deep AutoEncoder. Furthermore, the steps from 5 to 22 describe the outlier handling process of BIRCH [13] clustering algorithm. As a result, ODC handles the outlier in the network traffic data well than the existing deep clustering combinations. The evaluation of resultant clusters of ODC is discussed in SectionV.

V. EXPERIMENTAL EVALUATION

An enhanced deep AutoEncoder is implemented in Python using Keras [42]. Experiments on our datasets are conducted on a regular laptop with the Intel Core i7 processor. To evaluate our algorithm ODC, we use CoAP off-path dataset [5] to find out the anomalies in IoT network traffic and the standard publicly available MNIST [43] image dataset to compare the accuracy of ODC results with other existing works. We use the testbed from [5] to get more instances of IoT traffic with a CoAP off-path attack and feed the proposed algorithm with 10,000 unlabeled instances of IoT-CoAP traffic. We are ready to give the CoAP off-path dataset if anyone wants to redo the experiment for their research. To the best of our knowledge, our work is the first to combine deep AutoEncoder with the BIRCH clustering algorithm for anomaly detection in IoT network traffic datasets. The MNIST dataset has 70,000 digits of 28 × 28 pixels. We use publicly released codes by the respective DEC and IDEC authors to execute the corresponding algorithms to our dataset.

The encoder of our ODC contains two hidden layers and an input layer for both the datasets MNIST and CoAP off-path, as in Figure.1. The decoder part contains two hidden layers and an output layer for both the datasets MNIST and CoAP off-path. The dimension of the encoder is set as input data dimension(d) - 1626 - 756 - 50. The decoder dimension is set as a reverse of the encoder, such as 50 - 756 - 1626 - output dimension(d). The graphs in Figure. 2 and Figure. 3 shows that the reconstruction error rate fall depends on the choice of having activation and optimization functions. Though the relu, adamax combination, and ELU, dropout combinations with baseline deep AutoEncoder have a similar

Algorithm 1 Optimized deep clustering with BIRCH for outlier handling

Input: Input data: X; Epoch: E; Batch-size: I; Branching factor: B; Threshold: T;
Output: Label s
Data: Training/Testing set x
Parameters: Optimized deep auto-encoder weight W, Cluster radius R and Cluster centers

```
Let t=0
while iter < E do
    if iter I == 0 then
        Compute latent points h_i = f_θ(x_i) = σ(Wx_i + b) by applying (11) (9) and (13)
        Start CF tree t1 as in IV-B of initial T
        Continue scanning data and insert into t1
        if out of memory then
            Increase T
            Rebuild CF tree t2 of new T from CF tree t1
            if leaf data point of t1 is an outlier and disk space available then
                Write that data point as outlier
                else
                    use the data point to rebuild t2
            if t1 <= t2 then
                if Disk has space then
                    Go to step 5 and repeat the process for the rest of the data points
                    else
                        Re-absorb potential outliers into t1
                        Go to step 5 and repeat the process for the rest of the data points
    else
        Re-absorb potential outliers into t1
        Go to step 5 and repeat the process for the rest of the data points
```

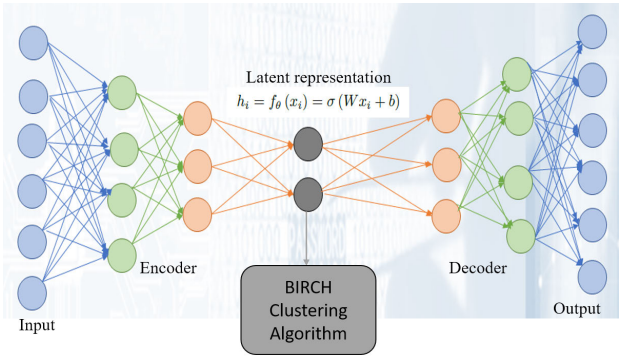

FIGURE 1. Enhanced deep clustering (enhanced deep AutoEncoder + BIRCH clustering).

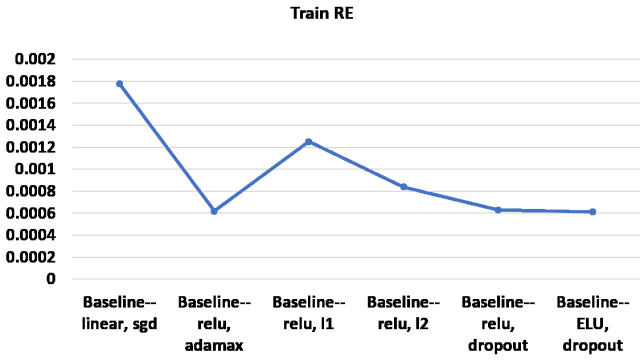

FIGURE 2. Comparing the training reconstruction error (Train RE) of deep AutoEncoder with various combinations of activation and optimization functions.

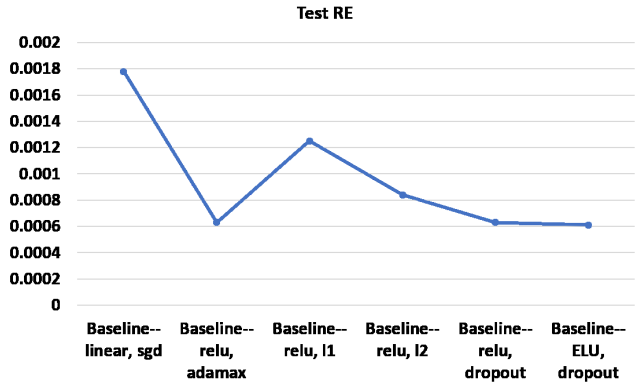

FIGURE 3. Comparing the testing reconstruction error (Test RE) of deep AutoEncoder with various combinations of activation and optimization functions.

reconstruction error rate, the later combination (ELU, dropout) produces consistent low reconstruction error rate for different iterations and different datasets. At the time of training the model, the decoder is used to reduce the reconstruction error rate. Once the model is optimized with a low reconstruction error, we merge the BIRCH clustering technique with the encoder's latent representation.

The clustering accuracy (ACC) depends on the branching factor (B) and the threshold value (T). When training the clustering algorithm, we choose the value of B and T through several iterations. We start to set the value of B as 15 and T as 1.5 to get good clustering accuracy and NMI. Branching factor value and Threshold value influence the ACC and NMI of the CoAP off-path dataset. It is noted that, when the threshold value and branching factor value decreases, we get the good ACC and NMI value, as shown in the graphs of Figure. 8 and Figure. 9. Hence, B and T values are directly proportional to the ACC and NMI values of a dataset.

The Table 2 shows our proposed algorithm ODC has the highest clustering accuracy than the state-of-the-art deep clustering methods. The method mentioned in Table 2, AE (AutoEncoder) with the k-means algorithm performs the k-means clustering algorithm on the latent representation

TABLE 2. Comparing the accuracy of various deep clustering techniques.

Method	MNIST		CoAP off-path	
	ACC	NMI	ACC	NMI
AE with K-means	81.82	74.73	82.93	75
DEC [8]	86.55	83.72	87.68	84.10
IDEC [9]	88.06	86.72	89.13	87.40
AE-l_2 with k-means [32]	90.20	87.10	91.04	87.00
DDC [10]	96.50	93.20	96.76	93.38
ODC-ours	**97.50**	**95.50**	**98.3**	**95.7**

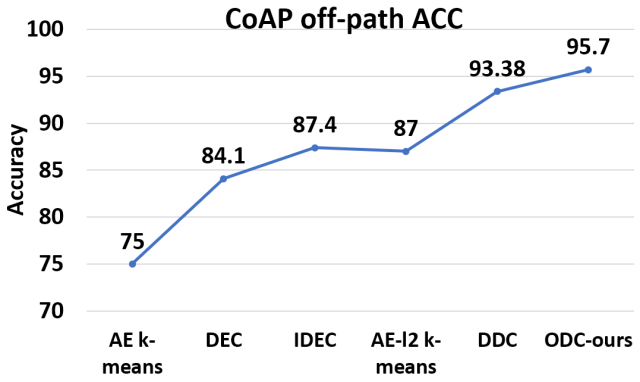

FIGURE 4. Accuracy of CoAP dataset.

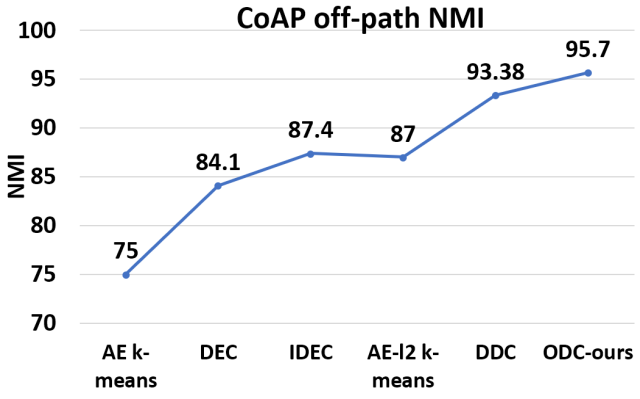

FIGURE 5. NMI of CoAP dataset.

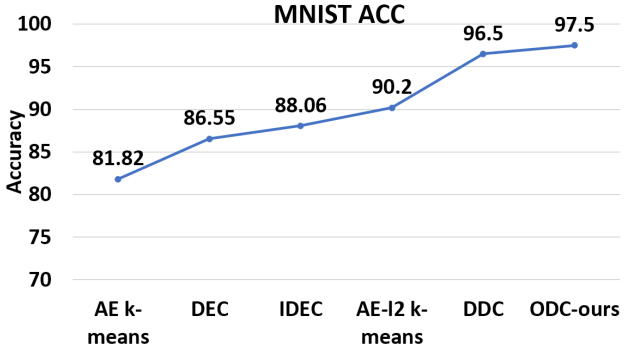

FIGURE 6. Accuracy of MNIST dataset.

of the trained AutoEncoder. We use the same AutoEncoder parameters as ours (ODC) to evaluate the AE+K-means deep clustering method.

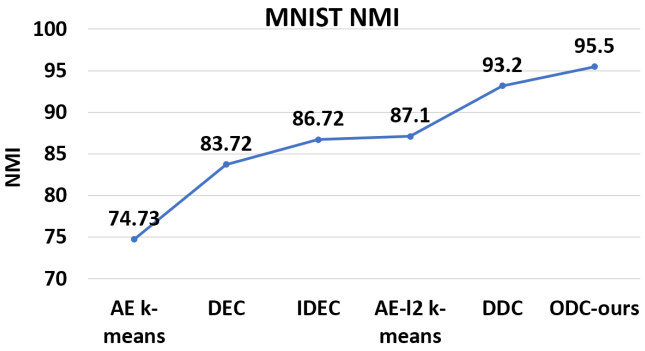

FIGURE 7. NMI of MNIST dataset.

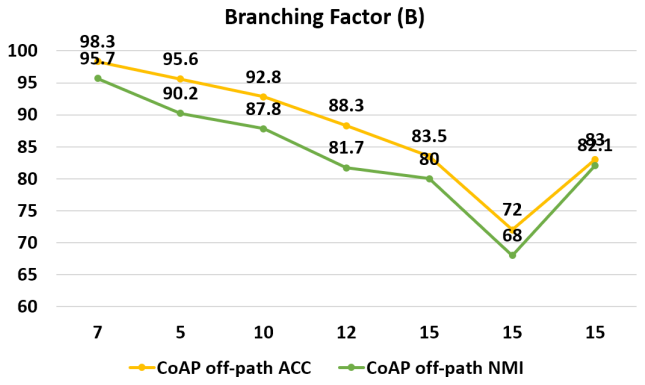

FIGURE 8. ACC and NMI of CoAP off-path dataset based on branching factor of BIRCH.

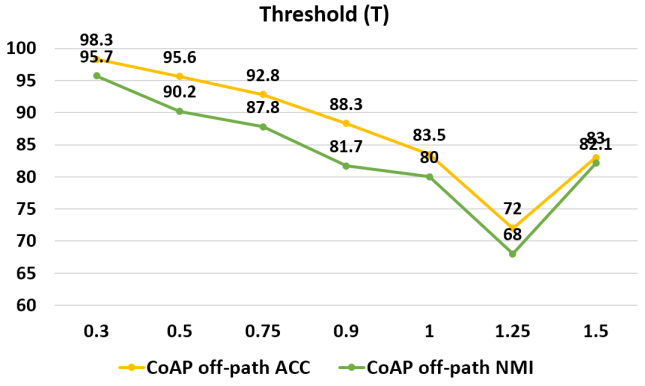

FIGURE 9. ACC and NMI of CoAP off-path dataset based on threshold value of BIRCH.

We utilize two standard unsupervised evaluation measurements for evaluation and correlations with the benchmark strategies, clustering Accuracy (ACC), and Normalized Mutual Information (NMI). ACC is defined as,

$$ACC = \max_{m} \frac{\sum_{i=1}^{n} \mathbb{I}\{l_i == m(c_i)\}}{n} \tag{14}$$

and NMI is defined as,

$$NMI = \frac{I(l;c)}{\max\{H(l); H(c)\}} \tag{15}$$

where the $\mathbb{I}\{\cdot\}$ is sign function, the l_i is the ground-truth label, c_i is the cluster assignment of the i_{th} sample predicted by the algorithm, and m ranges over all possible one-to-one mapping between predicted clusters and labels. $l = \{l_i\}_{i=1}^{n}$ $c = \{c_i\}_{i=1}^{n}$, respectively. n is the number of samples. $I(l; c)$ denotes the mutual information between l and c, and $H(\cdot)$ denotes their entropy. Both ACC and NMI are in [0, 1], and the higher scores imply more accurate clustering results. The graphs in Figure. 4, Figure. 5 and Figure. 6, Figure. 7 show, how well our proposed ODC algorithm outperforms the state-of-the-art deep clustering algorithms. For MNIST dataset, ODC achieves 0.975 ACC and 0.955 NMI. For CoAP off-path dataset, ODC accomplishes 0.983 ACC and 0.957 NMI.

VI. CONCLUSION

We proposed an intelligent anomaly detection algorithm ODC for extensive network traffic analysis. IoT environments produce a massive amount of data, and we need a mechanism/model to detect anomalies within the vast datasets. ODC optimizes the deep AutoEncoder to train the encoder. The latent version of the network traffic instances is fed into the BIRCH clustering algorithm for anomaly detection without human intervention. We demonstrated that ODC intelligently detects the anomalies for vast datasets. We analyzed B and T values' influence on the ACC and NMI values of an input dataset. The performance of the ODC deep clustering algorithm is evaluated through our implementation, and results presented in Table 2 clearly shows that our proposed scheme exhibits better performance in comparison with existing schemes.

Future directions of our work would be experimenting with ODC metrics other than Euclidean distance for anomaly detection. Our ODC anomaly detection method can be upgraded further by automating the whole anomaly detection model. This involves generating an alert message and send it to the system or network administrator without delay. Also, the suspected network traffic causing source can be identified and terminated or suspended from the regular network communication in a fraction of seconds.

REFERENCES

[1] A. Faigon, K. Narayanaswamy, J. Tambuluri, R. Ithal, S. Malmskog, and A. Kulkarni, "Machine learning based anomaly detection," U.S. Patent 10 270 788, Apr. 23, 2019.
[2] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 305–316.
[3] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi, "Real-time network anomaly detection system using machine learning," in *Proc. 11th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Mar. 2015, pp. 267–270.
[4] T. Dunning and E. Friedman, *Practical Machine Learning: A New Look at Anomaly Detection*. Sebastopol, CA, USA: O'Reilly Media, 2014.
[5] A. G. Roselin, P. Nanda, S. Nepal, X. He, and J. Wright, "Exploiting the remote server access support of CoAP protocol," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9338–9349, Dec. 2019.
[6] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song, "Anomalous example detection in deep learning: A survey," *IEEE Access*, vol. 8, pp. 132330–132347, 2020.
[7] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long, "Deep learning for anomaly detection," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 894–896.
[8] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[9] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *Proc. IJCAI*, 2017, pp. 1753–1759.
[10] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," 2018, *arXiv:1812.04287*. [Online]. Available: http://arxiv.org/abs/1812.04287
[11] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 373–382.
[12] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with Gaussian mixture variational autoencoders," 2016, *arXiv:1611.02648*. [Online]. Available: http://arxiv.org/abs/1611.02648
[13] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996.
[14] W. Wang, D. Yang, F. Chen, Y. Pang, S. Huang, and Y. Ge, "Clustering with orthogonal AutoEncoder," *IEEE Access*, vol. 7, pp. 62421–62432, 2019.
[15] M. Zamini and G. Montazer, "Credit card fraud detection using autoencoder based clustering," in *Proc. 9th Int. Symp. Telecommun. (IST)*, Dec. 2018, pp. 486–491.
[16] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.
[17] N. Mrabah, N. M. Khan, R. Ksantini, and Z. Lachiri, "Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction," 2019, *arXiv:1901.07752*. [Online]. Available: http://arxiv.org/abs/1901.07752
[18] X. Peng, H. Zhu, J. Feng, C. Shen, H. Zhang, and J. T. Zhou, "Deep clustering with sample-assignment invariance prior," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4857–4868, Nov. 2020.
[19] X. Peng, J. Feng, S. Xiao, W.-Y. Yau, J. T. Zhou, and S. Yang, "Structured AutoEncoders for subspace clustering," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5076–5086, Oct. 2018.
[20] P. Harrington, *Machine Learning in Action*. Shelter Island, NY, USA: Manning Publications, 2012.
[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
[22] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental local outlier detection for data streams," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Mar. 2007, pp. 504–515.
[23] M. Salehi and L. Rashidi, "A survey on anomaly detection in evolving data: [With application to forest fire risk prediction]," *ACM SIGKDD Explor. Newslett.*, vol. 20, no. 1, pp. 13–23, May 2018.
[24] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A survey of anomaly detection techniques in financial domain," *Future Gener. Comput. Syst.*, vol. 55, pp. 278–288, Feb. 2016.
[25] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, "A survey of anomaly detection for connected vehicle cybersecurity and safety," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 421–426.
[26] S. Jose, D. Malathi, B. Reddy, and D. Jayaseeli, "A survey on anomaly based host intrusion detection system," *J. Phys., Conf. Ser.*, vol. 1000, no. 1, Apr. 2018, Art. no. 012049.
[27] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using SVM and GA," in *Proc. 6th Annu. IEEE Syst., Man Cybern. (SMC) Inf. Assurance Workshop*, Jun. 2005, pp. 176–183.
[28] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of Data Mining in Computer Security*. Boston, MA, USA: Springer, 2002, pp. 77–101.
[29] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Proc. IEEE Int. Conf. Commun.*, vol. 5, Jun. 2006, pp. 2388–2393.
[30] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*. [Online]. Available: http://arxiv.org/abs/1901.03407
[31] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 178–183.
[32] C. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and unsupervised anomaly detection with l_2 normalized deep auto-encoder representations," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–6.

[33] Y. Wang, Z. Shi, X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep embedding for determining the number of clusters," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–2.
[34] J. Ding, A. Condon, and S. P. Shah, "Interpretable dimensionality reduction of single cell transcriptome data with deep generative models," *Nature Commun.*, vol. 9, no. 1, pp. 1–13, Dec. 2018.
[35] M. Kriegerowski, G. M. Petersen, H. Vasyura-Bathke, and M. Ohrnberger, "A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms," *Seismol. Res. Lett.*, vol. 90, no. 2A, pp. 510–516, Mar. 2019.
[36] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, Nov. 2018.
[37] S. M. Mousavi, W. Zhu, W. Ellsworth, and G. Beroza, "Unsupervised clustering of seismic signals using deep convolutional autoencoders," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 11, pp. 1693–1697, Nov. 2019.
[38] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. H. Hoi, and Z. Xu, "Semi-supervised deep embedded clustering," *Neurocomputing*, vol. 325, pp. 121–130, Jan. 2019.
[39] Y. Gu, X. Lu, L. Yang, B. Zhang, D. Yu, Y. Zhao, L. Gao, L. Wu, and T. Zhou, "Automatic lung nodule detection using a 3D deep convolutional neural network combined with a multi-scale prediction strategy in chest CTs," *Comput. Biol. Med.*, vol. 103, pp. 220–231, Dec. 2018.
[40] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: http://arxiv.org/abs/1511.07289
[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[42] A. Gulli and S. Pal, *Deep Learning With Keras*. Birmingham, U.K.: Packt, 2017.
[43] *THE MNIST DATABASE of Handwritten Digits*. Accessed: Feb. 4, 2020. [Online]. Available: http://yann.lecun.com/exdb/mnist/

ANNIE GILDA ROSELIN received the B.Tech. (Hons.) degree in information technology from the Noorul Islam College of Engineering, Anna University, Chennai, India, and the M.E. (Hons.) degree in computer science and engineering from the Mepco Schlenk Engineering College, Sivakasi, Anna University, Chennai, India. She is currently pursuing the Ph.D. degree with University of Technology Sydney. She has worked as a Lecturer with the Department Computer Science and Engineering, Velammall Engineering College, Chennai, from 2008 to 2011. Her research interest includes end-to-end IoT security: authentication, vulnerability exploration, and data analysis with deep learning. She is the successful recipient of the Industrial Scholarship funded by CSIRO/Data61, Australia.

PRIYADARSI NANDA received the bachelor's degree in computer engineering, the master's degree in computer engineering, and the Ph.D. degree in computing science. He is currently a Senior Lecturer with the University of Technology Sydney (UTS) with extensive experience in research and development of cyber security, the IoT security, and wireless sensor network security. His most significant work has been in the area of intrusion detection and prevention systems using image processing techniques, Sybil attack detection in IoT-based applications, intelligent firewall design. He has successfully supervised ten research students in the past and currently supervising eight research students in Cyber security research. He has published more than 80 high quality refereed research articles, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Future Generation Computer Systems* (FGCS) as well as many ERA Tier A/A* conference papers. In 2017, his work in cyber security research has earned him and his team the prestigeous Oman Research Council's National Award for best research.

SURYA NEPAL received the B.E. degree from the National Institute of Technology, Surat, India, the M.E. degree from the Asian Institute of Technology, Bangkok, Thailand, and the Ph.D. degree from RMIT University, Australia. He is currently a Principal Research Scientist with the Information Engineering Laboratory, CSIRO Computational Informatics. He has more than 15 years experience in computer science research, latterly with a specific focus on security, privacy and trust in distributed systems. He has more than 100 publications to his credit, has edited or coauthored several books, and is the co-inventor of two patents. Much of his work appears in top international forums, such as VLDB, ICDE, ICWS, SCC, CoopIS, ICSOC, the *International Journal of Web Services Research*, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *ACM Computing Survey*, and *ACM Transactions on Internet Technology*.

XIANGJIAN HE (Senior Member, IEEE) is currently a Professor of computer science with the School of Electrical and Data Engineering, University of Technology Sydney (UTS), and a Core Member, Global Big Data Technologies Associate Member with the AAI—Advanced Analytics Institute. As a Chief Investigator, he has received various research grants, including four national Research Grants awarded by the Australian Research Council (ARC). He is also the Director of the Computer Vision and Pattern Recognition Laboratory, Global Big Data Technologies Centre (GBDTC), UTS. He is also a Leading Researcher in several research areas, including big-learning based human behaviour recognition on a single image, image processing based on hexagonal structure, authorship identification of a document, and a documents components, such as sentences, and sections, network intrusion detection using computer vision techniques, car license plate recognition of high speed moving vehicles with changeable and complex background, and video tracking with motion blur. He has been a member with the IEEE Signal Processing Society Student Committee. He has been awarded the Internationally Registered Technology Specialist by the International Technology Institute (ITI).