© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Emergent Deep Learning for Anomaly Detection in Internet of Everything

Youcef Djenouri, Djamel Djenouri, Asma Belhadi, Gautam Srivastava**, and Jerry Chun-Wei Lin*

***Abstract*—This research presents a new generic deep learning framework for anomaly detection in the Internet of Everything (IoE). It combines decomposition methods, deep neural networks, and evolutionary computation to better detect outliers in IoE environments. The dataset is first decomposed into clusters, while similar observations in the same cluster are grouped. Five clustering algorithms were used for this purpose. The generated clusters are then trained using Deep Learning architectures. In this context, we propose a new recurrent neural network for training time series data. Two evolutionary computational algorithms are also proposed: the genetic and the bee swarm to fine-tune the training step. These algorithms consider the hyper-parameters of the trained models and try to find the optimal values. The proposed solutions have been experimentally evaluated for two use cases: 1) road traffic outlier detection and 2) network intrusion detection. The results show the advantages of the proposed solutions and a clear superiority compared to state-of-the-art approaches.**

***Index Terms*—Internet of Everything, Intrusion Detection, Smart Transportation, Deep Learning.**

I. INTRODUCTION

In this research work, we focus on the new offshoot of the Internet of Things (IoT), the Internet of Everything (IoE). The IoE extends the IoT by placing a greater emphasis on machine-to-machine (M2M) communication to describe more complex systems that can include people and processes, while considering intelligent connectivity and data processing. This concept enables the accumulation of an enormous amount of data. Effective processing and analysis of such Big Data, while challenging, will drive innovative applications in various fields such as cloud services [1], smart healthcare [2], smart buildings [3], robotics [4], and others. Anomaly detection refers to the process of filtering out anomalies from collected data. The term anomaly is general and can be used to refer to many problems, depending on the application erroneous data that may occur due to faulty sensors or during the data fusion process [5], road traffic outliers, or computer network intrusions [6], [7]. This research work is in this direction and proposes a new intelligent framework to efficiently and accurately identify anomalies in IoE environments.

Y. Djenouri is with the Dept. of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway, youcef.djenouri@sintef.no

D. Djenouri is with the CSRC, Dept. of Computer Science and Creative Technologies, University of the West of England, Bristol, UK, djamel.djenouri@uwe.ac.uk

A. Belhadi is with the Dept. of Technology, Kristiania University College, Oslo, Norway, asma.belhadi@kristiania.no

G. Srivastava is with the Dept. of Mathematics & Computer Science, Brandon University, Canada, and Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan, srivastavag@brandonu.ca (**Co-corresponding author)

J. C. W. Lin is with the Dept. of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway, jerrylin@ieee.org (*Corresponding author)

Most current anomaly detection solutions in IoE [6]–[8] are time consuming and have low accuracy. Deep learning based solutions [6], [7] provide relatively better accuracy compared to traditional solutions [8], but the improvement is still limited. The main reason is that they need to build a complex model with a high number of parameters to be specified. For example, the recurrent neural network (RNN) [9] requires a large number of states, and each state has parameters that need to be set. Evolutionary computation [10] is also widely studied for anomaly detection, but these solutions are limited only by exploration of the observation space and evaluate each observation separately. Motivated by the success of decomposition, deep learning (DL) and evolutionary computation in solving many real-world applications [11], [12], this research proposes a hybrid framework for inferring anomalies from IoE.

In this paper, we propose deep learning-based decomposition and evolutionary computation framework for anomaly detection networks (D2E-ADN) that aims to build targeted learning models for inferring anomalies in IoE. The data collected from the IoE environment is first divided into several small but as independent clusters as possible, minimizing the number of shared data between the clusters. The generated clusters are used to train the DL models, with each cluster used to train its own model. A hyperparameter optimizer is also investigated to accurately find the relevant parameters of the DL models. In this sense, the main contributions of this work are as follows:

1) We propose five decomposition algorithms for clustering data while extracting the relevant features from the IoE. The data clusters are then identified using clustering algorithms whose goal is to minimize the number of the shared data between clusters.
2) We propose a new DL model that uses the knowledge gained in the decomposition step. It is based on the recurrent neural network developed for processing time series data.
3) We propose two evolutionary computational algorithms to tune the parameters of the different steps of the D2E-ADN system, including the number of clusters in the decomposition step, the number of epochs, the learning error rate, and the activation functions for the DL models. The first evolutionary computational algorithm explores genetic optimization, while the second considers the behavior of the bees in exploring the possible configuration of the hyperparameters of the

This is the author accepted version of an article published in
IEEE Internet of Things Journal
http://dx.doi.org/10.1109/JIOT.2021.3134932

D2E-ADN system.

4) We evaluate D2E-ADN by comparing its computation time and accuracy with basic anomaly detection algorithms in two areas: intelligent transport (detecting outliers in traffic flow) and network security (detecting intrusions). This evaluation shows that D2E-ADN outperforms the baseline algorithms in both runtime and accuracy.

We give an outline of the remainder of this paper here. Section II gives an in-depth literature survey of existing solutions in anomaly detection. Next, in Section III, we present our proposed approach detailing all of its main components. Section IV gives our experimental analysis, discussion, and results. Lastly, Section V terminates our paper with some closing ideas.

II. Related Work

Zhong *et al.* [6] proposed a hybrid DL model for intrusion detection in a large network. The set of relevant features is first extracted using the damped incremental statistics algorithm. Then, the autoencoder algorithm is implemented to generate the training data, which is finally used to train the recurrent neural network model. Pawar *et al.* [13] proposed a DL framework for intrusion detection in the context of video-based activity recognition. An intensive comparative study of existing traditional machine learning techniques and advanced DL intrusion detection algorithms was conducted. Roberto *et al.* [7] developed a model for a convolutional neural network to identify abnormal traffic flows. The authors also provided a strategy for generating the labeled data used in the learning process. Khan *et al.* [14] proposed a novel two-stage DL algorithm for network intrusion detection. Network traffic is first classified into two classes (normal vs. abnormal) based on a probability score, which is then used as an additional feature to identify normal behavior or attack classes. Jallad *et al.* [15] used long-term memory (LSTM) to identify different types of intrusion detection such as point anomalies, collective anomalies, and contextual anomalies. The solution was tested on a large network for several million packets using the Spark platform. The results confirm the usefulness of the methods over traditional methods such as kNN.

Abdurrahman *et al.* [16] proposed a hybrid model that derives botnet in network. It combines convolutional networks and recurrent neural networks in the overall process. The relevant features are extracted based on a graph structure strategy. The extracted features are then converted into feature vectors and considered as training data for the hybrid recurrent neural convolutional network model. Garg *et al.* [17] developed a model (hybrid) using the Boltzmann machine, which has been constrained as well as the SVM (Support Vector Machine) in identifying abnormal activities in social media (multimedia) networks. The approach uses an incremental strategy and includes a self-learning mechanism where the anomalies already detected are fed into the DL model. Pektas *et al.* [16] combined the convolutional neural network and the LSTM using spatiotemporal features of network flows. Specifically, the convolutional neural network learns the spatial features of the network, while the long-term memory learns the temporal features. Ujjan *et al.* [18] presented an adaptive pooling-based sampling method to accurately infer distributed denial-of-service attacks in IoT. It integrates the snort intrusion detection system with the stacked autoencoder DL model to optimize detection accuracy in the control plane. Papamartzivanos *et al.* [19] developed a semi-supervised self-adaptive algorithm by integrating sparse autoencoder and feed-forward autoencoder to train the unlabeled data. Ferrag *et al.* [20] provided an overview of DL -based algorithms for detecting intrusions on 35 datasets. The DL models used in this study are based on neural networks (convolutional, recurrent), self-learning, and deep-belief networks. The detailed results show that the convolutional NN performs better than the models in both runtime and accuracy. Boukela *et al.* [21] developed the modified local outlier factor to mitigate the malfunction of security systems in IoT devices. This approach takes into account the handling of high-dimensional data, determining the reachability distance for all features of the selected neighbors. Edje *et al.* [22] developed a clustering-based algorithm for identifying fault and event outliers in IoT sensors. The event outliers are considered when there are problems in sensor readings. Noshouhi *et al.* [23] presented a new machine learning-based solution for predicting fires using spatiotemporal measurements. Relevant data such as temperature and humidity are trained, and the model attempts to separate abnormal cases from normal behaviors. A refinement process is also performed to ensure that the predicted anomalies are not due to outliers. Zhang *et al.* [24] seeks to ensure the confidentiality of Industrial Internet of Things customers by combining blockchain and federated learning. The fault detection system is developed to provide complete verification of customer data. Lin *et al.* [25] developed a multi-objective algorithm based on ant colony optimization metaheuristics for privacy preservation in IoT environment. The ant colony solution space is encoded and represented by hiding sensitive information. An external archive is used to preserve the extracted Pareto solutions. Chou *et al.* [26] proposed a taxonomy of intrusion detection datasets used for evaluation in the last two decades. In addition, future directions are proposed by extending intrusion detection to a cloud environment and creating ground truth based data in real network environments.

From this extensive literature review, it is clear that traffic anomaly detection solutions are often weak in terms of detection rate because the entire database must be considered during the learning process. Moreover, it is not clear how to tune the hyperparameters for DL models. In this work, we investigate a hybrid approach that combines PSO, decomposition, and CNN to efficiently find outliers and anomalies in traffic databases. We use both cluster-based algorithms and swarm-based approaches to tune CNN.

III. Deep learning-based Decomposition and Evolutionary computation for Anomaly Detection Network

A. Principle

Here, we present the proposed D2E-ADN framework that integrates decomposition, DL, and evolutionary computational

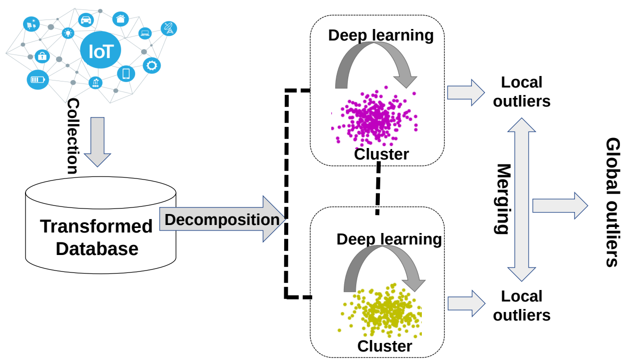

Fig. 1. Illustration of the D2E-ADN framework

optimization to identify anomalies in the data environment. As shown in Fig. 1, the AD2E-ADN consists of three steps: i) decomposition, which divides the data into clusters such that each cluster contains similar data. ii) DL model, whose goal is to apply the DL process to each cluster to identify local anomalies. A merging strategy is used to combine the local anomalies into global anomalies. iii) Evolutionary computation, which is used to learn the hyperparameters of the models of the clusters. In the following, each step is explained in detail.

B. Decomposition

The main required aim to this step is for dividing the whole data into k clusters, $C = \{C_1, C_2, \dots, C_k\}$, where each cluster $C_s = \{D_1^{(s)}, D_2^{(s)}, \dots, D_{|C_s|}^{(s)}\}$ is the subset of the data D. The overlapping data is minimized within clusters, and overlapping data in each and every cluster is maximized. In other words, using Eq. 1:

$$\begin{cases} \arg\min_{C} | \bigcup_{i=1,j=1}^{k} ((C_i) \cap (C_j))|, i \neq j \\ \bigwedge \\ \arg\max_{C} | \bigcup_{C_s}^{C} (D_i^{(s)} \cap D_j^{(s)}) | \forall (i,j) \in [1..|C_s|]^2, i \neq j \end{cases} \tag{1}$$

It is necessary to use different clustering algorithms than in previous work [27]–[30] to minimize the number of shared data between clusters and maximize the number of shared data in each cluster. The following concepts should be introduced here:

1) **Similarity computation**. The distance measure between two data D_i and D_j is calculated by subtracting the number of shared items from the number of all items between D_i and D_j, as given in Eq. 2.

$$Dist(D_i, D_j) = \max(|D_i|, |D_j|) - (|D_i \cap D_j|) \tag{2}$$

2) **Centroids updating**. Here, we should consider datasets of each cluster $C_i = \{D_1^{(i)}, D_2^{(i)}, \dots, D_{|C_i|}^{(i)}\}$, the aim is to find a gravity center of this set which is also a datum. The centroid, μ_i, is computed based on the centroid formula developed in [31]. Each item's frequency can be calculated for all the data in a cluster C_i. Data center length given as l_i is connected to the avg. number of datum within C_i, as shown in Eq. 3.

$$l_i = \frac{\sum_{j=1}^{|C_i|} |D_j^{(i)}|}{|C_i|} \tag{3}$$

Afterwards, the data within C_i can be sorted by frequency, and the frequency datum l_i is then assigned to μ_i, as $\mu_i = \{j | j \in l_i\}$.

3) **data neighborhoods**. Data neighbourhoods of D_i, denoted as $\mathcal{N}_{D_i}$, are defined by the set of all observations that are similar to D_i with a given threshold ϵ. It is computed as shown in Eq. 4.

$$\mathcal{N}_{D_i} = \{D_j | Dist(D_i, D_j) \leq \epsilon \vee j \neq i\} \tag{4}$$

4) **Core data**. Datum D_i is known as core data if and only if here is some minimum number of data σ_D, such that $|\mathcal{N}_{D_i}| \geq \sigma_D$.

5) **Shared data determination**. Upon the construction of the clusters of data, the shared set has to be determined of data between clusters. In Eq. 5, the shared sets for data denoted as S, are defined.

$$S = \bigcup_{i=1, j>i}^{k} C_i \cap C_j, \tag{5}$$

where $S^{i,j}$ is the shared set between clusters C_i and C_j.

1) Naive grouping for data decomposition: For naive groupings, the main aim is to be able to group data into k clusters that are disjoint without the need for any processing. With m datum, $\{D_1, D_2, \dots, D_m\}$, the first $\frac{m}{k}$ datum are assigned to C_1, the second $\frac{m}{k}$ to C_2, and so until assigning all that datum to the k clusters.

2) Hierarchical agglomerative clustering for data decomposition: HAC (Hierarchical Agglomerative Clustering) [27] for data decomposition which has the main aim in the creation of tree-like nested structure partitions, $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_h\}$, of the data such that, $\forall (i,j) \in [1, \dots, k]^2, \forall (m,l) \in [1, \dots, h]^2, C_i \in \mathcal{H}_m, C_j \in \mathcal{H}_l, m \geq l \Rightarrow C_i \in C_j \wedge C_i \cap C_j = \emptyset$. First, there is a starting point with all data points in separate clusters. Next, we keep connecting two clusters that can be agreed to be very similar until we reach the point of a single cluster. We can define the similarity between any two clusters C_i and C_j by determining the number of common elements between them, or $|C_i \cap C_j|$.

3) K-means for data decomposition: We know that K-means [28] is trying to optimize the function: $J = \sum_{j=1}^{k} \sum_{D' \in C_j} |D' - \mu_j|^2$, where μ_j is the centroid of the data in C_j. A centroid is computed for each cluster, and then the data are randomly distributed among k clusters. Then, each datum is assigned to a cluster based on which centroid is closest to it. These steps are repeated until no more assignments to clusters are made, at which point the procedure terminates itself.

4) Bisecting k-means for data decomposition: In the bisecting k-means algorithm, when [29] decomposes the data, it does so using both hybrid partitioning and a divisive hierarchical methodology. We start with a single cluster and then split a cluster into 2 in each individual step, using the standard k-means approach. Looking more closely at the approach, the process of bisecting clusters can be repeated many times, with higher similarity achieved in the division.

5) DBSCAN for data decomposition: In the DBSCAN algorithm [30], the main goal in data decomposition is to be able to search for clusters in each ϵ neighborhood per datum. Once the core data is found, DBSCAN is responsible for iteratively collecting all density-reachable data directly from the core data. This process may result in some density reachable clusters being merged individually. We can stop the process if no new data is added to a cluster.

C. DL model

Here is presented a new DL model for detecting anomalies in data. It is based on a recurrent neural network and considers time series as input. The input of the recurrent neural network is the set of clusters generated in the previous step. As a result, different models are generated, each of which is associated with a data cluster. Our model network is a (many-to-many) architecture. The problem of the model is binary classification, i.e., outputting a class label indicating whether the data is anomalous or not. This is done for each datum in the cluster. A multilayer feedforward network is applied to each data cluster, consisting of multiple neurons arranged in layers. Each neuron of layer l is connected to each neuron of layer $(l-1)$ with a certain weight. Each input datum D_{i-1} is connected to a group of neurons in the input layer. The neurons in the output layer are associated with the output of the model (the class label 1 for anomalous or 0 for normal). The goal is to reduce the error between the output data of the model and the ground truth of the data, such as:

$$E(D) = \sum_{i=1}^{|D|} E(D_i), \tag{6}$$

where,

$$E(D_i) = \sqrt{\sum_{j=1}^{|D_{ij}|} (D_{ij} - \widehat{D_{ij}})^2)} \tag{7}$$

The output of the m^{th} neuron in the layer l, noted o_l^m is given by Eq. 8. Note that the sum of the outputs of all neurons in the given layer should be between 0 and 1. Here, we have the following equations as:

$$o_l^m = \sigma(\sum_{j=1}^{|l-1|} o_{l-1}^j \omega_{l-1}^{mj} + b_l^m), \tag{8}$$

with

$$\sum_{m=1}^{|l|} o_l^m = 1, \tag{9}$$

where $\sigma(.)$ is the activation function, $|l|$ is the number of neurons in the layer l, o_{l-1}^j is the output of the j^{th} neuron in the $l-1$ layer, ω_{l-1}^{mj} is the weight value that connects the neurons o_l^m and o_{l-1}^j, and b_l^m is the bias value associated to the neuron o_l^m.

At each iteration i, the updating weight rule is given as by:

$$\omega_{l-1}^{mj}(i) = \omega_{l-1}^{mj}(i-1) - \mu \times D_i \times 2 \times E_i, \tag{10}$$

where μ is the learning parameter rate, and,

$$E_i = \sum_{j=1}^{|D_i|} (D_{ij} - \hat{D_{ij}})^2 \tag{11}$$

At the end of the learning step, different models will be designed, and one for each cluster, C_i. We define a local ranking vector $Rank_i$ by applying a learning model M_i on the cluster C_i, denoted $Rank_i = M_i(C_i)$. The process of the global ranking of the data D is performed as follows:

1) Compute the score of each D_j, say $Score(D_j)$.
2) Sort the scores of the data, D, in an ascending order.
3) Retrieve the top anomalous according to the scores of D.

D. Evolutionary Computation

In this section, we can show the process by which we can determine the optimal set for the D2E-ADN approach to finding the set of hyperparameters. Here we can define a set of hyperparameters given by $\mathcal{HP} = \{\mathcal{HP}_1, \mathcal{HP}_2, \dots, \mathcal{HP}_r\}$. Here r is defined as the total number of hyperparameters. Each $\mathcal{HP}_i$ can be represented in a set of possible values for a given hyperparameter. Moreover, we define our configuration space $\mathcal{CS}$ such that we can say that the set of possible configurations where each configuration can be represented as a vector in the possible values for all hyperparameters $\mathcal{HP}$. Thus, the hyperparameter problem for optimization has the main goal of finding an optimal configuration that provides the highest accuracy for both the regression and classification rates. We can also say that the size of the configuration space can depend on the number of possible values of the hyperparameters. We can use Eq. 12 such that:

$$|\mathcal{CS}| = \prod_{i=1}^{r} |\mathcal{HP}_i|. \tag{12}$$

Here we can clearly see that the configuration space can be very large. For example, if only $1,000$ possible values per epoch parameter and 100 per error rate and $1,000$ for the number of bounding boxes (i.e. CNN) are considered, then the configuration space could be as large as 100 million. Therefore, we need to be able to avoid exhaustive search approaches as they are inappropriate for this type of problem. To solve this problem, evolutionary computational algorithms need to be explored. In the following, we discuss the main components of such approaches.

1) Population Initialization: Considering the initial population represented as pop_size, the individuals must be distributed over the configuration space $\mathcal{CS}$. This allows exploration of different configurations and coverage of most regions in $\mathcal{CS}$. When generating the initial population, we can start the process by generating a random individual that can represent a configuration $\mathcal{CS}$. This individual can then generate $pop\_size - 1$ individuals, keeping in mind that each new individual can be dissimilar to the already generated individuals. The dissimilarity of two configurations can be easily determined by the distance between the configurations of the individuals in question. We can also say that the initial population, given as $\mathcal{P}$, should be able to maximize the diversification function using the Eq. 13.

$$Diversify(\mathcal{P}) = \sum_{i=1}^{|\mathcal{P}|} \sum_{j=1}^{|\mathcal{P}|} Distance(\mathcal{CS}_i, \mathcal{CS}_j), \quad (13)$$

where we note here that $Distance(\mathcal{CS}_i, \mathcal{CS}_j)$ is defined as the distance between i^{th}, and j^{th} individuals configurations, respectively.

2) Crossover: For the generation of any new offspring, we must ensure that the steps as follows are applied:

- A crossover point is generated at random which ranges from 1 to r, creating a *left side* and *right side* split.
- *left side* of first individual can be transferred to *left side* of first offspring. However, *right side* of first individual can be copied to *right side* of second offspring.
- *left side* for second individual can be copied to *left side* for second offspring. Moreover, *right side* of second individual can be copied to *right side* of first offspring.

3) Mutation: The diversification of the search is increased by a mutation operation. By itself, the technique consists only in randomly changing the parameter values for each configuration. Once a random mutation point has been generated, which can range from 1 to r, future mutation point values can be generated using the crossover operator.

4) Local Search: The local search tool starts with the individuals of the population and returns the neighbors. The neighbors are defined by updating the number of a parameter to the current setting. This process is repeated for all individuals of the population, with a high number of repetitions.

5) Fitness Function: As mentioned earlier, the D2E-ADN approach aims to jointly maximize the regression and classification ratios. With this in mind, a multicriteria function is proposed to be used when evaluating individuals from the populations as in Eq. 14.

$$Fitness(\mathcal{CS}_i) = \frac{\alpha \times CR(\mathcal{CS}_i) + \beta \times RR(\mathcal{CS}_i)}{2}. \quad (14)$$

We note here that,

- $\mathcal{CS}_i$ can be defined as the configuration of i^th individual in population.
- $CS(\mathcal{CS}_i)$ can be defined as the classification ratio of D2E-ADN algorithm using $\mathcal{CS}_i$.
- $RR(\mathcal{CS}_i)$ can be defined as the regression ratio of D2E-ADN algorithm using $\mathcal{C}_i$. We note here that $RR(\mathcal{CS}_i)$ can be set to 0 for RNN use.
- α and β can be defined as 2 user parameters that are set between 0.0 and 1.0.

Using the above operations, 2 algorithms are proposed for the hyperparameter optimization methods. In the first case, a genetic approach is used, and in the second case, a swarm optimization method is used. It is shown that both approaches are efficient when used with large populations.

TABLE I
PERCENTAGE (%) OF THE SHARED DATA OF THE CLUSTERING STEP FOR THE D2E-ADN FRAMEWORK

Dataset	naive grouping	HAC	kmeans	bisecting kmeans	DBSCAN
Odense	42	40	5	7	30
Beijing	40	39	9	11	31
ICSX2012	39	37	7	18	24
CICIDS2017	45	31	8	10	21

6) Genetic Algorithm: The initial population of individuals of size pop_size is first randomly generated. Each individual is constructed with respect to the initialization of the population. Then, the crossover, mutation, and local search operators are applied to generate configurations from $\mathcal{CS}$. To maintain the same size of the population, all individuals are evaluated using the fitness function and only the first pop_size individuals (in terms of quality) are left while the others are removed. The identical procedure is continued until the predefined maximum number of iterations is reached.

TABLE II
DETECTION RATIO OF THE DL STEP FOR THE D2E-ADN FRAMEWORK

Dataset	Epochs 100	Epochs 1,000	Epochs 10,000
Odense	0.65	0.70	0.70
Beijing	0.70	0.72	0.72
ICSX2012	0.70	0.73	0.73
CICIDS2017	0.71	0.72	0.72

TABLE III
FITNESS COMPUTING OF THE EVOLUTIONARY COMPUTATION STEP FOR THE D2E-ADN FRAMEWORK

Dataset	Genetic Algorithm	Bees Swarm Optimization
Odense	0.78	0.79
Beijing	0.77	0.80
ICSX2012	0.80	0.79
CICIDS2017	0.81	0.79

7) Bees Swarm Optimization Algorithm: First, a bee searches for a good feature configuration. After this initial configuration is found, a set of configurations *SearchArea* in the search space using Eq. 13. Each individual particle viewed from the *SearchArea* is the starting point for the search. After a local search process is complete, each individual bee passes its "best visited" configuration to all neighboring bees using a table known as *Dance*. In the *Dance* table, a stored configuration then becomes the next reference for the next iteration. To ensure that no cycles occur, each new reference configuration is added to a tab list that must never be used as a starting reference again. If, after several iterations, it is determined that the swarm does not improve its configuration, the diversification criterion is introduced to avoid trapping the local optimum. Usually, the diversification criterion consists of a distant configuration that is not stored in the tabu list. The algorithm usually ends when the optimal version is found or a maximum number of iterations is reached.

IV. Experimental Evaluation

Several experiments were conducted to validate the usefulness of the proposed framework using two real case studies. The first is urban traffic anomalies used in intelligent transportation and the second is intrusion detection for securing World Wide Web technologies. Evaluation measures include detection accuracy using the F-measure [32] and runtime. All experiments were implemented on a $128 - bit$ Core i9 processor with UBUNTU 20 and $32GB$ from RAM used in conjunction with a GPU device, an NVIDIA Tesla C2086 with 534 CUDA cores (16 multiprocessors with 64 cores each) and a clock speed of $2.15GHz$. There is $3.2GB$ of global memory, $59.15KB$ of shared memory, and a warp size of 64. Both the CPU and GPU use single precision.

A. Datasets

Urban Traffic Anomaly Detection: Two real urban traffic datasets were used: i) The first was obtained from Odense Municipality (Denmark)[1]. This is a set of lines containing information about the detection of cars and their locations. The flows were observed between 1^{st} January 2017 and 30^{th} April 2018 and consist of more than 12 million cars and bicycles. ii) The second one is from the Beijing traffic flow and was retrieved from Beijing City Lab[2]. It consists of more than 900 million traffic flow values during two months in one place. The anomalies in these two datasets are the set of traffic flows, which may be a single traffic value or a sequence of traffic values in a given time window.

Intrusion Detection: Many intrusion detection datasets, such as KDD and DARPA, have been widely used over the past two decades. However, these datasets are outdated and do not reflect current security attacks in modern computer networks, which are characterized by the emergence of IoT-generated traffic. The ISCX2012 [3] data were recently generated to reflect current attack scenarios on networks. They consist of seven days of real malicious and normal network activity. The normal network traffic is generated by normal operations, while the attack scenarios are performed with human assistance to minimize misunderstandings with normal network traffic. There are four different attack options such as penetrating the network from inside, Hypertext Transfer Protocol Denial of Service, Distributed Denial of Service using botnets and Brute Force Secure Shell. The second data used is CICIDS2017 [33], which contains labeled network flows in CSV format. They were collected over a five-day period and include some cutting-edge attack scenarios such as brute force file transfer protocol, brute force secure shell, denial of service attack, web attack, infiltration, and botnet.

B. D2E-ADN Parameter Setting

1) Decomposition: The first experiment aims to evaluate, on different datasets, the quality of the following decomposition algorithms: intuitive grouping, HAC, k-means, bisecting

[1]https://www.odense.dk/
[2]https://www.beijingcitylab.com/
[3]http://www.unb.ca/cic/datasets/index.html.

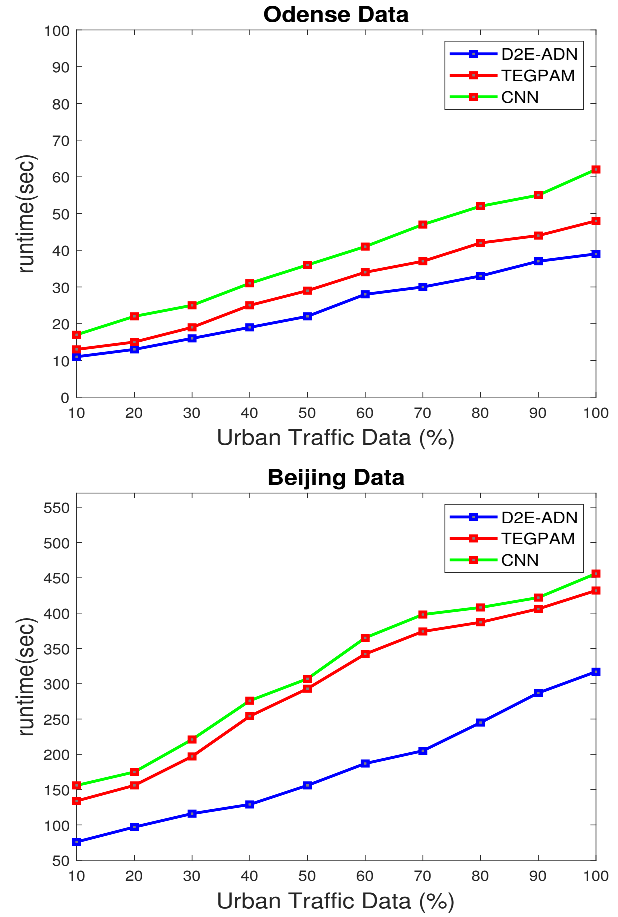

Fig. 2. Runtime in seconds of D2E-ADN versus state-of-the art urban traffic anomaly detection algorithms

k-means and DBSCAN. This is determined by the percentage of separation data between clusters/groups, while high quality is reflected by low values of this percentage. The number of clusters was varied from 1 to 50 for the Naive Grouping and k-means algorithms, and the ϵ value was varied from 1 to 10 for the DBSCAN algorithm. In this experiment, the optimal parameter values for each clustering method are used and are shown in Table I. Note that the number of clusters 5 for intuitive clustering, 7 for k-means and bisecting k-means, 12 for HAC, and ϵ for DBSCAN was set to 4. The number of separation data with the best parameter values for each database is presented. The results show that k-means and bisecting k-means provide better decomposition into records compared to the other three algorithms. These results can be explained by the fact that k-means and bisecting k-means are pure partitioning, i.e., both algorithms are oriented to the centroids representing the data of the same cluster. DBSCAN, on the other hand, is inspired by computing neighborhoods to represent dense regions. Consequently, it is conceivable that two datasets are comparable and belong to the two closest clusters. In the following tests, we use the k-means decomposition technique of our framework.

2) Performance of DL Model: Here, we are concerned with computing the quality of the DL step of 1) the convolutional neural network for urban traffic anomaly detection and 2) the

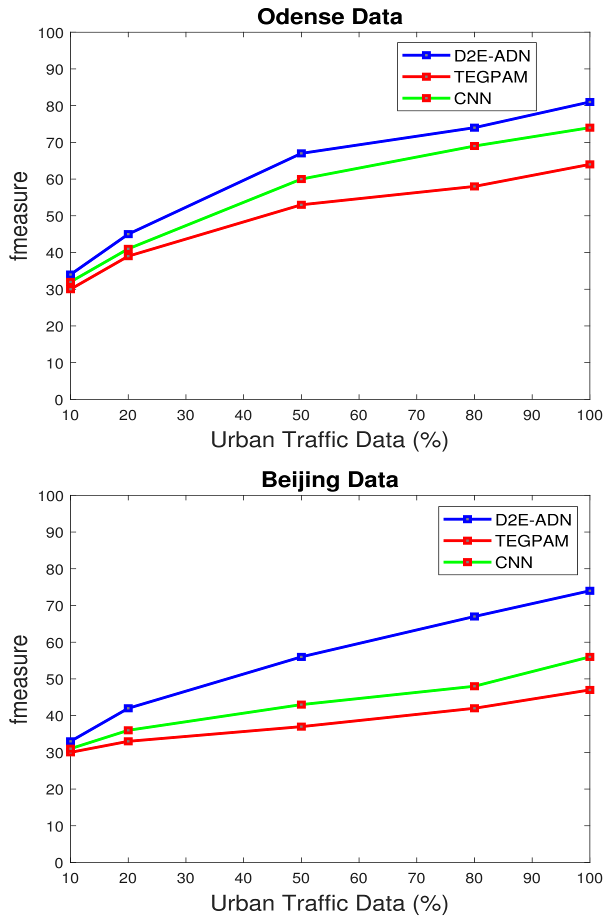

Fig. 3. Accuracy of D2E-ADN versus the state-of-the art urban traffic anomaly detection algorithms

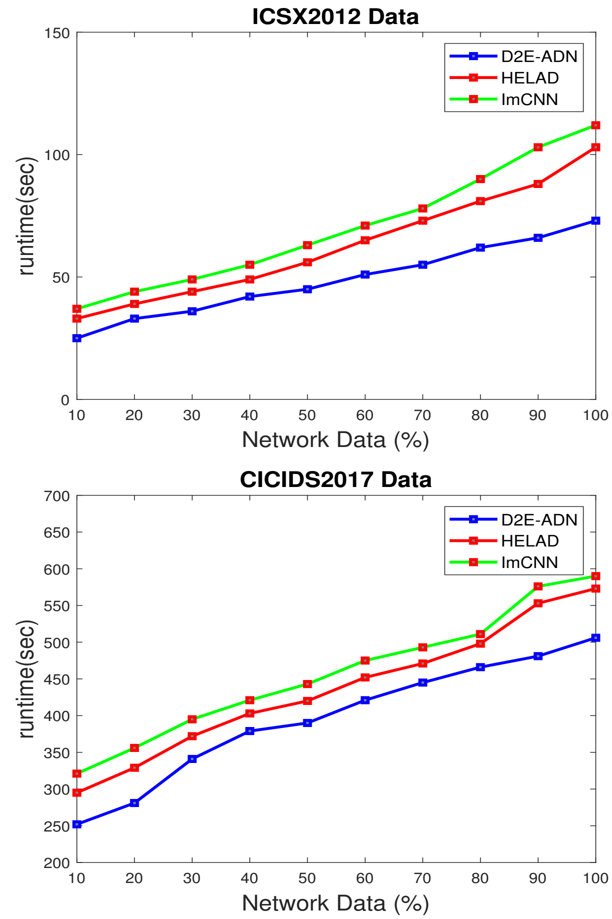

Fig. 4. Runtime in seconds of D2E-ADN versus the state-of-the art intrusion detection algorithms

recurrent neural network for intrusion detection. The quality is determined by the detection rate, which is the ratio between the number of detected outliers and the number of all outliers. If you vary the number of epochs of the network from 100 to $10,000$, Table II shows that the detection rate of both algorithms increases up to $1,000$ and then converges at this value. The reason for these results is that the weights of both models became stable after $1,000$ iterations. Therefore, the best epochs for both algorithms are $1,000$, which is used in the rest of the experiments.

3) Evolutionary Computation: In this part, the quality of the evolutionary computational step in genetic algorithms and bee swarm optimization is evaluated. This quality is determined by the best value of the fitness calculation of the final population.

By varying the number of individuals/bees from 1 to 100 and the maximum number of iterations from 1 to 100, the best parameter values for each evolutionary computation algorithm are used in this experiment and listed in Table III. Note that the number of individuals and the maximum number of iterations are 35 and 47, respectively, for the genetic algorithm, while the number of bees and the maximum number of iterations are 43 and 59, respectively, for the swarm optimization algorithm. The results show that the genetic algorithm is better for intrusion detection and the bee swarm optimization is better for urban anomaly detection. In the remaining experiments, we used the genetic algorithm for intrusion detection and the bee swarm optimization for urban traffic anomaly detection.

C. Results for Urban Traffic Anomaly Detection

In this experiment, we compare the performance of the D2E-ADN algorithm with TEGPAM [8], and CNN [34], as baseline urban traffic anomaly detection algorithms.

1) Runtime: In Fig. 2, the running time in seconds of D2E-ADN is shown in comparison to the baseline algorithms. It shows that the running time of the three algorithms increases with the percentage of data. For 10% of data, all algorithms require less than 200 seconds to identify outliers and more than 350 seconds to process the entire data. The results also show the superiority of our approach compared to the other two algorithms, with a difference of more than 100 seconds for processing the entire data. These results were obtained thanks to the efficient combination of the convolutional neural network with the decomposition algorithms in deriving anomalies from the urban traffic data.

2) Accuracy: In Fig. 3, the F-measure of the D2E-ADN is shown in comparison with the baseline algorithms. It shows that the F-measure increases with the percentage of data in the three algorithms. Most importantly, it shows the clear superiority of D2E-ADN with an advantage of more than 15 points in processing the whole data. These results are obtained thanks to the efficient combination of the convolutional neural network with the evolutionary computation in the optimization of the hyperparameters. Thus, finding the appropriate parameters for learning the network can significantly improve the detection rate of outliers.

D. *Results for Intrusion Detection*

This part compares D2E-ADN with HELAD [6] and ImCNN [35], as two baseline algorithms for network intrusion detection.

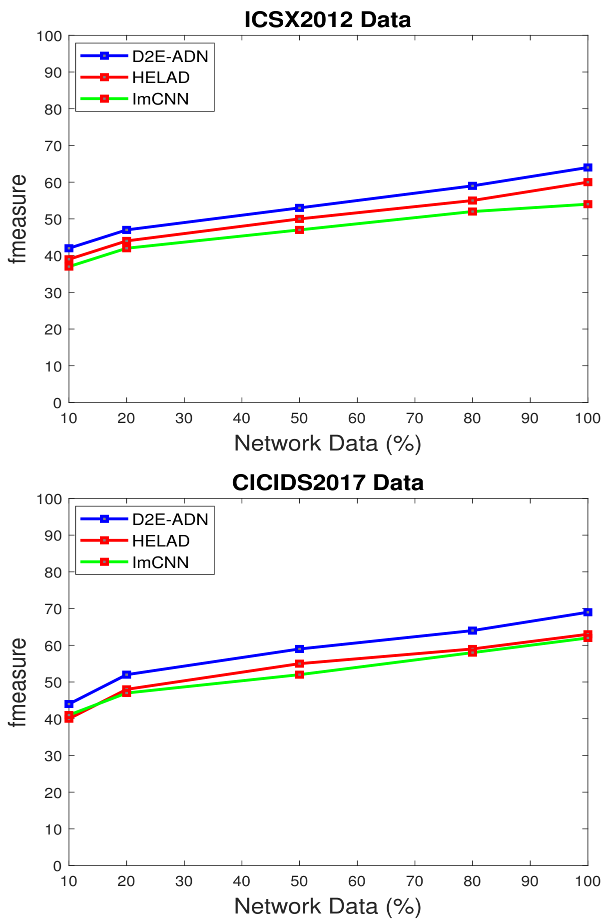

Fig. 5. Accuracy of D2E-ADN versus state-of-the art intrusion detection algorithms

1) Runtime: Fig. 4 shows the runtime in seconds of D2E-ADN, HELAD, and ImCNN on ICSX2012 and CICIDS2017 datasets. The results show that the runtime of the three algorithms increases with the percentage of data. This has significant implications, e.g., all algorithms require less than 250 seconds to identify an anomaly from 10% of the data, but more than 550 seconds to process the entire data. The results also show the superiority of the proposed approach (D2E-ADN) compared to the other two algorithms, with a difference of more than 150 second for processing the whole data. These results are obtained thanks to the efficient combination of the recurrent neural network with the decomposition algorithms in deriving anomalies from the urban traffic data. Any RNN that learns from homogeneous data can significantly increase the performance in detecting outliers.

2) Accuracy: The F-measure of D2E-ADN compared with the baseline algorithms (HELAD and ImCNN) is shown in Fig. 5. The results show that the F-measure of the three algorithms increases with the percentage of data. They also reveal the superiority of D2E-ADN, which offers the advantage of more than 12 points for processing the whole data. These results are obtained thanks to the efficient combination of the recurrent neural network with the evolutionary computation in the optimization of the hyperparameters of our algorithm.

V. Conclusion

In this work, we studied the problem of anomaly detection in IoE and proposed a combination of decomposition, deep neural networks and evolutionary computation to find anomalies from the dataset. In our approach, the dataset is first decomposed into similar clusters using different types of clustering algorithms. The clusters are then trained using an extended recurrent neural network. To perform the training step efficiently, two evolutionary computation algorithms are proposed to take the hyper-parameters of the trained models and try to find the optimal ones. Several experiments in the form of two case studies for two different IoE applications show the advantages of the proposed solution compared to the basic approaches. In perspective, we plan to explore other data representations such as trajectories. We also plan to propose a parallel version that explores high-performance computing to increase the performance of the proposed solution and train the data clusters simultaneously. In addition, the current work can be extended to other subsets of the digital IoT world. Although IoE is a recent development, other areas within IoT can be explored using the concepts presented in this paper. For example, both the Internet of Vehicles (IoV) and the Internet of Smart Infrastructures (III) could be a future home for the research presented here. In this context, in addition to the datasets used here, other novel datasets can be used to further test and refine the work already done.

Acknowledgment

This research was partially funded by the Natural Sciences Research Council of Canada (NSERC) Discovery Grant program (RGPIN-2020-05363) held by Dr. Gautam Srivastava.

References

[1] Y. Miao, X. Liu, K. R. Choo, R. H. Deng, H. Wu, and H. Li, “Fair and dynamic data sharing framework in cloud-assisted internet of everything,” *IEEE Internet Things Journal*, vol. 6, no. 4, pp. 7201–7212, 2019.

[2] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, “Internet of things (iot): A review of its enabling technologies in healthcare applications, standards protocols, security and market opportunities,” *IEEE Internet of Things Journal*, 2021.

[3] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, “Machine learning for smart building applications: Review and taxonomy,” *ACM Computing Surveys*, vol. 52, no. 2, pp. 24:1–24:36, 2019.

[4] B. Ouyang, P. S. Wills, Y. Tang, J. O. Hallstrom, T.-C. Su, K. Namuduri, S. Mukherjee, J. I. Rodriguez-Labra, Y. Li, and C. J. Den Ouden, “Initial development of the hybrid aerial underwater robotic system (haucs): Internet of things (iot) for aquaculture farms,” *IEEE Internet of Things Journal*, 2021.

[5] S. Boulkaboul and D. Djenouri, "DFIOT: data fusion for internet of things," *J. Netw. Syst. Manag.*, vol. 28, no. 4, pp. 1136–1160, 2020.

[6] Y. Zhong, W. Chen, Z. Wang, Y. Chen, K. Wang, Y. Li, X. Yin, X. Shi, J. Yang, and K. Li, "Helad: A novel network anomaly detection model based on heterogeneous ensemble learning," *Computer Networks*, vol. 169, p. 107049, 2020.

[7] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, 2020.

[8] L. Lin, J. Li, F. Chen, J. Ye, and J.-P. Huai, "Road traffic speed prediction: A probabilistic model fusing multi-source data," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[9] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles." in *The International Joint Conference on Artificial Intelligence*, 2019, pp. 2725–2732.

[10] A. Karale, M. Lazarova, P. Koleva, and V. Poulkov, "A hybrid pso-milof approach for outlier detection in streaming data," in *The International Conference on Telecommunications and Signal Processing*. IEEE, 2020, pp. 474–479.

[11] Y. Djenouri, A. Belhadi, P. Fournier-Viger, and J. C. W. Lin, "Fast and effective cluster-based information retrieval using frequent closed itemsets," *Information Sciences*, vol. 453, pp. 154–167, 2018.

[12] A. Belhadi, Y. Djenouri, J. C.-W. Lin, C. Zhang, and A. Cano, "Exploring pattern mining algorithms for hashtag retrieval problem," *IEEE Access*, vol. 8, pp. 10 569–10 583, 2020.

[13] K. Pawar and V. Attar, "Deep learning approaches for video-based anomalous activity detection," *World Wide Web*, vol. 22, no. 2, pp. 571–601, 2019.

[14] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30 373–30 385, 2019.

[15] K. Al Jallad, M. Aljnidi, and M. S. Desouki, "Big data analysis and distributed deep learning for next-generation intrusion detection system optimization," *Journal of Big Data*, vol. 6, no. 1, p. 88, 2019.

[16] A. Pektaş and T. Acarman, "A deep learning method to detect network intrusion through flow-based features," *International Journal of Network Management*, vol. 29, no. 3, p. e2050, 2019.

[17] S. Garg, K. Kaur, N. Kumar, and J. J. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: A social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, 2019.

[18] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sflow and adaptive polling sampling for deep learning based ddos detection in sdn," *Future Generation Computer Systems*, 2019.

[19] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13 546–13 560, 2019.

[20] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[21] L. Boukela, G. Zhang, M. Yacoub, S. Bouzefrane, S. B. B. Ahmadi, and H. Jelodar, "A modified lof-based approach for outlier characterization in iot," *Annals of Telecommunications*, vol. 76, no. 3, pp. 145–153, 2021.

[22] A. E. Edje, S. M. Abd Latiff, and H. W. Chan, "Enhanced non-parametric sequence-based learning algorithm for outlier detection in the internet of things," *Neural Processing Letters*, vol. 53, no. 3, pp. 1889–1919, 2021.

[23] M. R. Nosouhi, K. Sood, N. Kumar, T. Wevill, and C. Thapa, "Bushfire risk detection using internet of things: An application scenario," *IEEE Internet of Things Journal*, 2021.

[24] W. Zhang, Q. Lu, Q. Yu, Z. Li, Y. Liu, S. K. Lo, S. Chen, X. Xu, and L. Zhu, "Blockchain-based federated learning for device failure detection in industrial iot," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5926–5937, 2020.

[25] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, "Privacy-preserving multiobjective sanitization model in 6g iot environments," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5340–5349, 2020.

[26] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–36, 2021.

[27] W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of classification*, vol. 1, no. 1, pp. 7–24, 1984.

[28] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, 1967, pp. 281–297.

[29] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD Workshop on Text Mining*, vol. 400, no. 1. Boston, 2000, pp. 525–526.

[30] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of KDD*, 1996, pp. 226–231.

[31] Y. Djenouri, D. Djamel, and Z. Djenoouri, "Data-mining-based decomposition for solving MAXSAT problem: Towards a new approach," *IEEE Intelligent Systems*, 2017.

[32] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 427–438.

[33] A. Pektaş and T. Acarman, "Classification of malware families based on runtime behaviors," *Journal of information security and applications*, vol. 37, pp. 91–100, 2017.

[34] L. Zhu, R. Krishnan, A. Sivakumar, F. Guo, and J. W. Polak, "Traffic monitoring and anomaly detection based on simulation of luxembourg road network," in *IEEE Intelligent Transportation Systems Conference*, 2019, pp. 382–387.

[35] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.