Cyber-Typhon: An Online Multi-task Anomaly Detection Framework

Konstantinos Demertzis[1(✉)], Lazaros Iliadis[1], Panayiotis Kikiras[2], and Nikos Tziritas[3]

[1] School of Engineering, Department of Civil Engineering, Faculty of Mathematics Programming and General Courses, Democritus University of Thrace, Kimmeria, Xanthi, Greece
kdemertz@fmenr.duth.gr, liliadis@civil.duth.gr
[2] School of Science, Department of Computer Science, University of Thessaly, Lamia, Greece
kikirasp@uth.gr
[3] Research Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
nikolaos@siat.ac.cn

Abstract. According to the Greek mythology, Typhon was a gigantic monster with one hundred dragon heads, bigger than all mountains. His open hands were extending from East to West, his head could reach the sky and flames were coming out of his mouth. His body below the waste consisted of curled snakes. This research effort introduces the "Cyber-Typhon" (CYTY) an Online Multi-Task Anomaly Detection Framework. It aims to fully upgrade old passive infrastructure through an intelligent mechanism, using advanced Computational Intelligence (COIN) algorithms. More specifically, it proposes an intelligent Multi-Task Learning framework, which combines On-Line Sequential Extreme Learning Machines (OS-ELM) and Restricted Boltzmann Machines (RBMs) in order to control data flows. The final target of this model is the intelligent classification of Critical Infrastructures' network flow, resulting in Anomaly Detection due to Advanced Persistent Threat (APT) attacks.

Keywords: Deep content inspection · Anomaly detection · Multi-task learning · Online learning · Restricted Boltzmann Machine · Critical Infrastructure Protection

1 Introduction

Information generated by complex environments such as the Internet of Things (IOT) ecosystem, has increased exponentially. The result is the inefficient management and storage of the total volume of generated information. This requires the adoption of complex data mining and analysis architectures [1]. These architectures should incorporate specialized processing algorithms, that dynamically adapt to new standards or data, or to scaled data production as a equation of time [2].

Published by Springer Nature Switzerland AG 2019
J. MacIntyre et al. (Eds.): AIAI 2019, IFIP AICT 559, pp. 19–36, 2019.
https://doi.org/10.1007/978-3-030-19823-7_2

Although mining Data Streams (DAST) is an emerging area, it poses enormous challenges to the data mining community. High transmission speed, change of data distribution, and high volume, raise the following issues that should be addressed [3]:

- *High Velocity*: Online DAST, arrive at a very high speed. Thus, it has become almost practically infeasible to scan all of them. This is also the case for the offline ones.
- *Concept Drift*: Frequent Patterns (FREP) keep changing, as data streams are time varying in nature. During the mining process, as new incoming data is added to the existing ones, some FREP may change their status to become infrequent and vice versa. This issue is known as the Concept Drift (CDR) problem.
- *Unbounded Size* (UNS): DAST are unbounded in size. Their size is unknown to the user in advance unlike the static data.
- *Enormous Space Requirement* (ESR): Huge amount of data are generated both in online and offline applications. There might not be enough space to store the data stream before processing.
- *Unsteady Analysis Results* (UAR): High speed as well as varying data distribution, may affect the analyzed results, due to which the mining outcome may be declined. In order to cope with this, DAST mining must be an incremental process.

Anomaly Detection [4] over multiple data streams, is initially determined by the observation of a single (multi-variable) time series frequency, which constitutes the systems' quantitative performance parameters.

A data stream s_i may be coming from a system of sensors, and it consists of numerical values, where $s_i(t)$ stands for the data stream flow value at the time t, and $t \in [0, +\infty]$. If sensors' flows are synchronized periodically to report their values, the whole of the multi-variable information at each time t is represented with the following frame vector shown in Eq. 1, [5].

$$\Delta\Pi_t = (s_1(t), s_2(t), \ldots, s_n(t) \in \mathrm{R}^n) \tag{1}$$

In practice, each flow forms a one-dimensional time series, while the frame vector flow (FVF) represents a multi-variable time series. Event detection on DAST is intended to determine the values $si(t)$, which represent abrupt changes within an FVF.

Particularly, each FVF of length is converted to a binary vector of the same length, where each value represents a possible change in the corresponding sensor flux. Such deviations from the normal behavior are called events, and binary vectors are called event vectors. An event may be an observation that does not conform to an expected standard in the data set (anomaly). Incidents may have been caused by a variety of reasons, like sensor failure or malequation, or deviations and substantial changes that may affect the system's behavior, such as Cyber Attacks (CYA) [7]. Therefore, a vector of events in time t is represented by Eq. 2 below [3, 5].

$$\Delta\Sigma_t = \left(e_1^t, e_2^t, \ldots, e_n^t\right) \in [0, 1] \tag{2}$$

where $e_i^t = e_i(t)$ is the binary value which represents the occurrence of an abnormal flow behavior, and its value is $s_i(t) = 1$ in time t. The values of $s_i(t)$ must be in an interval $\{0, 1\}$.

The error is calculated at each iteration, as data characteristics can change drastically and in an unpredictable way changing the typical, normal behavior. An object that may be considered abnormal, can then be included in the set of normal observations due to rapid developments in the data stream. Due to the fact that the data volume is unlimited, data mining is performed on a subset of the flow, called a sliding window, which obviously contains a small but recent percentage of the observations. The goal of the data flow processing algorithms is to minimize the cumulative error for all iterations, that can be calculated by the following Eq. 3 [3, 5].

$$I_n[w] = \sum_{j=1}^{n} V\left(\langle w, x_j\rangle, y_j\right) = \sum_{j=1}^{n}\left(x_j^T w - y_j\right)^2 \tag{3}$$

where $x_j \in R^d, w \in R^d$ and $y_j \in R$. We consider that $Xi \times d$ is a data matrix and $Yi \times 1$ is a matrix with target values, after the arrival of the first i data points. Assuming that the covariance matrix $\Sigma_i = X^T X$ is reversible, the optimal solution $f^*(x) = \langle w^*, x\rangle$ is given by the following Eq. 4 [3, 5].

$$w^* = \left(X^T X\right)^{-1} X^T \Upsilon = \Sigma_i^{-1} \sum_{j=1}^{i} x_j y_j \tag{4}$$

First the covariance table is calculated by using the following Eq. 5

$$\Sigma_i = \sum\nolimits_{j=1}^{i} x_j x_j^T \tag{5}$$

The initial time complexity (CM) is calculated to be of $O(id^2)$ order, but after we inverse the $X^T X$ $(d \times d)$ table, it increases to $O(d^3)$, while the rest of the required multiplications have an $O(d^2)CM$. This produces an overall CM of $O(id^2 + d^3)$ [3, 5] (d is the size of the window).

It is conceivable that robust systems ensuring reliability and high accuracy rates without requiring high availability of resources are required to safely approach problems stemming from knowledge mining processes. The above argument is further supported as follows: Let's suppose that the size of data points is equal to n and after the arrival of each new data point i = 1, 2, …, n, the recalculation of the solution is required. In this case the total time complexity would be equal to $O(n^2 d^2 + nd^3)$ [5].

The Multi-Task Learning (MTL) is a robust method in order to face some of the most challenge of the Big Data Streams processing with Online Learning algorithms. MTL is a subfield of Machine Learning (ML) in which multiple learning tasks are solved at the same time, utilizing common elements or differences arising from the multiple tasks included in the case study [6, 7]. More general MTL is an inductive transfer method which generates many generalization features. The common features or differences that arise between distributed tasks during the training process are

transferred or shared as guaranteed and unambiguous knowledge in subsequent relevant or unrelated tasks, maximizing the accuracy of the model. MTL is efficient because regularization induced by requiring an algorithm to perform well on a related task can be superior to regularization that prevents overfitting by penalizing all complexity uniformly. The following approaches are characteristic cases of MTL [6, 7]:

- *Task grouping and overlapping,* where tasks are grouped or provided in an overlapping way so that there is relevance, capable to lead in the use of cognitive or learning relationships.
- *Exploiting unrelated tasks,* where the common learning of non-relevant tasks using the same input data, can be beneficial for learning main tasks of an application field.
- *Transfer of knowledge*, where transfer of relevant knowledge is carried out to achieve learning from correspondingly trained models.
- *Group online adaptive learning,* where transfer of previous experience or knowledge into continually changing environments takes place.

2 Literature Review

Chen and Abdelwahed [8] have applied autonomous computing technology, to monitor SCADA system performance. Their approach proactively estimates upcoming attacks for a given system model of a physical infrastructure. Soupionis et al. [9] have proposed a combinatorial method for automatic detection and classification of faults and cyber-attacks occurring on the power grid system when there is limited data from the power grid nodes due to cyber implications.

In addition, Zhu et al. [10] have described the network attack knowledge, based on the theory of the factor expression of knowledge. They have studied the formal knowledge theory of SCADA network from the factor state space and equivalence partitioning. This approach utilizes the Factor Neural Network (FNN) theory which contains high-level knowledge and quantitative reasoning, used to establish a predictive model including analytic FNN and analogous FNN. This model abstracts and builds an equivalent and corresponding network attack and a defense knowledge factors system.

Finally [11] has introduced a new European Framework-7 project Cockpit CI (Critical Infrastructure) and roles of intelligent machine learning methods to prevent SCADA systems from cyber-attacks. Also, existing multi-task learning methods can be categorized into two main categories: learning with feature covariance and learning with task relations [12]. Different from prior solutions to distributed multi-task learning, which are focused on the former category [13], our proposed multi-task learning falls into the latter category. On the other hand, distributed machine learning has attracted more and more interests recently [14].

There have been tremendous efforts done on different machine learning problems. Also, online multi-task learning assumes instances from different tasks arrive in a sequence and adversarial chooses task to learn. Cavallanti et al. [15] exploited online multi-task learning with a given task relationship encoded in a matrix, which is known

beforehand. Parallel multi-task learning aims to develop parallel computing algorithms for multi-task learning in a shared-memory computing environment.

Recently, Zhang [7] proposed a parallel multi-task learning algorithm named PMTL. In this work, dual forms of three losses are presented and accelerated proximal gradient method is applied to make the problem decomposable, and thus possible to be solved in parallel. Finally, distributed multi-task learning is an area that has not been much exploited.

Wang et al. [13] proposed a distributed algorithm for multi-task learning by assuming that different tasks are related through shared sparsity. In another work [6], asynchronous distributed multi-task learning method is proposed for multi-task learning with shared subspace learning or shared feature subset learning. Different from the above-mentioned approaches, our method aims at solving multi-task learning by learning task relationships from data, which can be positive, negative, or unrelated, via a task-covariance matrix.

3 The Proposed Cyber-Typhon Framework

This research proposes the Cyber-Typhon model, which combines the algorithmic power of OS-ELM and RBM in a hybrid mode. It is an innovative Multi-Task Learning approach [16], that performs control of network traffic in Critical Infrastructures [17–19]. The final target is the detection of vulnerabilities that are usually due to APT attacks [20–22].

More specifically, the Cyber-Typhon initially exports features related to network traffic, which are used as input to an OS-ELM neural network. The OS-ELM has been trained with proper data, in order to be able either to classify traffic as normal or (in the opposite case) to identify the threat or the attack type. Obviously, it performs Multi-class classification in order to identify one of the following eight (8) classes: Normal, Naïve Malicious Response Injection (NMRI), Complex Malicious Response Injection (CMRI), Malicious State Command Injection (MSCI), Malicious Parameter Command Injection (MPCI), Malicious Equation Code Injection (MFCI), Denial of Service (DoS) and Reconnaissance (Recon).

If the network traffic is normal further communication is allowed. In the opposite case, the type of anomaly is determined and the data flow is redirected to a proper absolutely specialized and dedicated Restricted Boltzmann Machine. If the first RBM does not recognize the specific anomaly for which it is specialized, the data is redirected to the next RBM responsible for the detection of another anomaly and so on till the successful identification is achieved. If detection cannot be done by any of the trained RBM (which are as many as the types of the known anomalies) the network flow data return to the initial OS-ELM, which can perform online sequential learning. Thus, the classification effort can be re-adjusted. The following Fig. 1 presents the architecture of the Cyber-Typhon.

The proposed ELM is a Single-Hidden Layer Feed Forward Neural Network (SHLF2N2) [23] with N hidden neurons, randomly selected input weights and random values of bias in the hidden layer. The output weights are calculated with a single vector matrix multiplication [13]. Hidden nodes or hidden level parameters can be

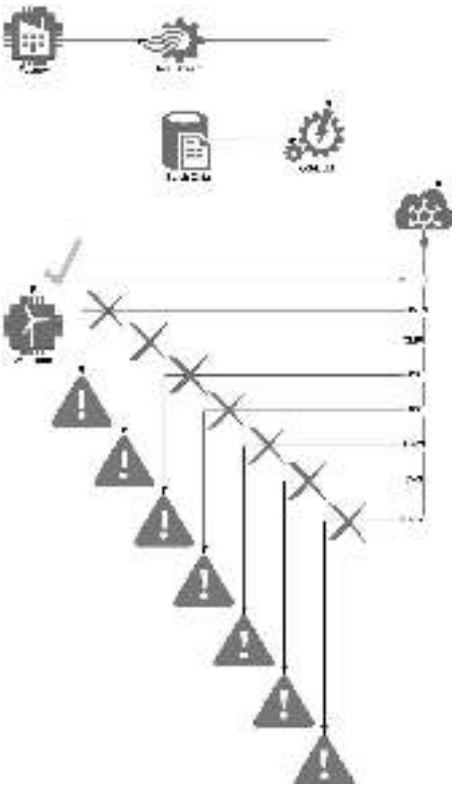

Fig. 1. Graphic representation of the Cyber-Typhon

randomly created before seeing the training data, while it is remarkable that non-differential activation equations can be handled, and known Neural Network problems such as stopping criterion, learning rate and learning epochs are not addressed. Specifically, the input data is mapped to a random L-dimensional space with a discrete training set N, where $(x_i, t_i), i \in [\![1, N]\!]$ with $x_i \in R^d$ and $t_i \in R^c$. The specification output of the network is given by the following Eq. 6 [23]:

$$f_L(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta \quad i \in [\![1, N]\!] \tag{6}$$

Vector matrix $\beta = [\beta_1, \ldots, \beta_L]^T$ is the output of the weight vector matrix connecting hidden and output nodes. On the other hand, $h(x) = [g_1(x), \ldots, g_L(x)]$ is the output of the hidden nodes for input x, and $g_1(x)$ is the output of the *ith* neuron. Based on a training set $\{(x_i, t_i)\}_{i=1}^{N}$, the ELM can solve the Learning Problem $H\beta = T$, where

$T = [t_1, \ldots, t_N]^T$ are the target labels and the output vector matrix of the Hidden Layer H is given by the following Eq. 7 [23]:

$$H(\omega_j, b_j, x_i) = \begin{bmatrix} g(\omega_1 x_1 + b_1) & \cdots & g(\omega_l x_1 + b_l) \\ \vdots & \ddots & \vdots \\ g(\omega_1 x_N + b_1) & \cdots & g(\omega_l x_N + b_l) \end{bmatrix}_{N \times l} \tag{7}$$

The input weight vector matrix of the hidden layer ω (before training) and the bias vectors b are created randomly in the interval [−1, 1], by employing Eqs. 8a and 8b.

$$\omega_j = \left[\omega_{j1}, \omega_{j2}, \ldots, \omega_{jm}\right]^T \tag{8a}$$

and

$$\beta_j = \left[\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}\right]^T \tag{8b}$$

The output weight vector matrix of the hidden layer H is calculated by the use of the activation equation in the training dataset, based on Eq. 9a and the output weights β are calculated based on Eq. 9b.

$$H = g(\omega x + b) \tag{9a}$$

$$\beta = \left(\frac{\mathrm{I}}{C} + H^T H\right)^{-1} H^T X \tag{9b}$$

where $H = [h_1, \ldots, h_N]$ is the output vector matrix of the hidden layer and $X = [x_1, \ldots, x_N]$ is the input vector matrix of the hidden layer. Indeed, β can be calculated by the following Eq. 10:

$$\beta = H^+ T \tag{10}$$

where H^+ is the generalized inverse Moore-Penrose vector for matrix H [23]. The Cyber-Tyfon employs the ELM algorithm, which employs the kernel of the following Gaussian Radial Basis Equation $K(u, v) = exp(-\gamma||u - v||^2)$ (5).

The number of the hidden neurons k is equal to 20 and the assigned random input weights are w_i where b_i, i = 1, …, N are the bias. The calculation of the hidden layer output matrix H has been done by employing Eq. 11 below [23].

$$H(h) = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_L(x_1) \\ \vdots & & \vdots \\ h_1(x_N) & \cdots & h_L(x_N) \end{bmatrix} \tag{11}$$

where $h(x) = [h_1(x), \ldots, h_L(x)]$ is the output (row) vector of the hidden layer with respect to the input x. Also, $h(x)$ maps the data from the D-dimensional input space to the L-dimensional hidden-layer feature space (ELM feature space) H. Thus, $h(x)$ is

indeed a feature mapping. ELM aims to minimize the training error $\|H\beta - T\|^2$ as well as the norm $\|\beta\|$ of the output weights:

where *H* is the hidden-layer output matrix of the Eq. 11.

Minimization of the norm of the output weights $||\beta||$ is actually achieved by maximizing the distance of the separating margins of the two different classes in the ELM feature space $2/||\beta||$. To calculate the output weights β we used Eq. 12 [23].

$$\beta = \left(\frac{I}{C} + H^T H\right)^{-1} H^T T \tag{12}$$

where the value of *C* (a positive constant) and the value of *T* are obtained from the *Equation Approximation* of the SHLF2N2 with additive neurons (see Eq. 13 below).

$$t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m, T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} \tag{13}$$

The OS-ELM [16] is an alternative technique for large-scale computing and machine learning approaches that used when data becomes available in a sequential order to determine a mapping from data set corresponding labels.

The main difference between online learning and batch learning techniques is that in online learning the mapping is updated after the arrival of every new data point in a scale fashion, whereas batch techniques are used when one has access to the entire training data set at once. It is a versatile sequential learning algorithm because of the training observations are sequentially (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm. At any time, only the newly arrived single or chunk of observations (instead of the entire past data) are seen and learned. A single or a chunk of training observations is discarded as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed. The learning algorithm has no prior knowledge as to how many training observations will be presented. Unlike other sequential learning algorithms which have many control parameters to be tuned, OS-ELM only requires the number of hidden nodes to be specified [23].

The proposed OS-ELM algorithm consists of two main phases namely:

The Boosting (BPh) and the Sequential Learning (SLPh).

The BPh is used to train the SLFNs by applying basic ELM, with a batch of training data in the initialization stage. The boosting training data are discarded as soon as BPh is completed. The volume of the required training data vectors is very small, and it can be equal to the number of hidden neurons. A detailed description of the OS-ELM classifier is done below, where Eqs. 9a and 9b are employed.

Phase 1 (BPh)

The BPh for a small initial training set $N = \left\{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \cdots, \widetilde{N}\right\}$ is the following:

(a) Assign arbitrary input weights w_i and biases b_i or centers μ_i and their corresponding impact widths σ_i, $i = 1, \cdots \widetilde{N}$, where $\widetilde{N}$ is the number of hidden neurons used by the RBF kernel for a specific application.

(b) Calculate the initial hidden layer output matrix $H_0 = \left[h_1, \cdots, h_{\widetilde{N}}\right]^{\mathrm{T}}$ where $h_i = \left[g(w_1 \cdot x_i + b_1), \cdots, g\left(w_{\widetilde{N}} \cdot x_i + b_{\widetilde{N}}\right)\right]^{\mathrm{T}}$, $i = 1, \cdots, \widetilde{N}$, and g is the activation equation or the RBF kernel.

(c) Estimate the initial output weight $\beta^{(0)} = M_0 H_0^T T_0$, where $M_0 = \left(H_0^T H_0\right)^{-1}$ and $T_0 = \left[t_1, \ldots, t_{\widetilde{N}}\right]^T$.

(d) Set $k = 0$.

Phase 2 (SLPh)

In the second SLPh, the OS-ELM algorithm learns the train data one-by-one, or chunk-by-chunk, and all of the training data are discarded once the learning procedure on them is completed. The actual steps of this phase (for each incoming observation (x_i, t_1)), are described below.

(a) Calculate the hidden layer output vector $h_{(k+1)} = [g(w_1 \cdot x_i + b_1), \cdots, g\left(w_{\widetilde{N}} \cdot x_i + b_{\widetilde{N}}\right)]^{\mathrm{T}}$.

(b) Calculate the latest output weight $\beta^{(k+1)}$ by employing the Recursive Least-Squares (RLS) algorithm where $\hat{\beta} = (H^T H)^{-1} H^T T$.

(c) Set $k = k + 1$, where $x_i \in R^n$, $t_i \in R^m$ and $i = \widetilde{N} + 1, \widetilde{N} + 2, \widetilde{N} + 3$: $x_i \in R^n$, $t_i \in R^m$ and $i = \widetilde{N} + 1, \widetilde{N} + 2, \widetilde{N} + 3$.

Obviously, if the network flow is classified as normal, it is allowed to continue. In the opposite case, there are 7 RBMs, as many as the abnormal classes, where each one of them has been trained to perform One-Class Classification (OCC) in order to exclusively recognize one specific network attack [24].

The OCC also known as Unary Classification Method (UCM), implements intelligent categorization of cases belonging to a specific class, among an existing set of records. OCC is learning from a training set that contains only records of one specific class. Usually in this method the negative class is absent because it is not sampled. Thus, the division boundaries are determined effectively only with the knowledge of the positive class.

This process is much more difficult than a traditional binary or multiclass classifier [25], as it is trained to accept target objects and to reject the ones that have significant deviation. Minimizing the errors is also a difficult process, because in this type of

categorization, cross validation is unavailable since there is no data from the other classes. Finally, it should be stressed that one class problem-solving technique is inverse to the generalization approaches that are pursued in other machine learning problems, as it tends to provide a fully defined configuration of parameters. This can exponentially increase the complexity of the classifier, trying to correctly classify target data. The more complex the model, the smaller the rank range in the target data range, and the less likely it is for the outliers to be categorized correctly. In practice, one can create a complex model by setting all its possible parameters without being at risk from overfitting.

In this sense, the OCC is the most appropriate approach for detecting abnormalities and identifying patterns or trends, in a set of data that displays divergent behaviors than expected. OCC achieves high levels of successful detection, while maintaining low false error rates (false alarm) [24, 25].

The RBM [26] belong to the family of energy-based models, where each configuration of the variables of interest corresponds to a finite scalar energy value used for training. The learning process is performed by modifying the energy equation (ENF), so that its shape has desirable.

Specifically, RBM is a symmetric graphical model. The units of one layer are connected (and thus dependent) only with units of the next one. The proposed RBM ENF with V visible units and H hidden ones is defined as follows Eq. 14 [26]:

$$E(v,h) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} v_i b_i^v - \sum_{j=1}^{V} h_j b_j^h \tag{14}$$

where v and h are binary vectors related to the state of visible units, and to the state of hidden units respectively. Moreover, v_i and h_j correspond to the individual state of each visible unit (VU) i, and each hidden unit (HU) j respectively, and w_{ij} is the weight assigned to their connection. Finally, b_i^v and b_j^h is the bias of the VU i and the HU j. The reserved probability $p(v|h)$ is given by the following Eq. 15:

$$p(v|h) = \frac{e^{-E(v,h)}}{\sum_g e^{-E(g,h)}} \tag{15}$$

In the specific case of examining a unit i of the visible level, the assigned reserved probability distribution (if we know the status of the hidden layer h), is calculated by Eq. 16:

$$p(v_k = 1|h) = \frac{1}{1 + e^{-\sum_{j=1}^{H} h_j w_{kj} + b_k^v}} \tag{16}$$

The reserved probabilities $p(h|v)$ and $p(h_k = 1|v)$ are defined by Eqs. 17 and 18:

$$p(h|v) = \frac{e^{-E(v,h)}}{\sum_g e^{-E(v,g)}} \tag{17}$$

and

$$p(h_k = 1|v) = \frac{1}{1 + e^{-\sum_{i=1}^{V} v_i w_{kj} + b_k^h}} \tag{18}$$

These relations express the independence of the units of the two levels from the units of the same level. The training process of the RBM is the process of finding values for its parameters, that maximize the mean logarithmic probability of the occurrence of set C (Eq. 19).

$$\sum_{c=1}^{C} \log_p \frac{\sum_g e^{-E(v^c,g)}}{\sum_u \sum_g e^{-E(u,g)}} \tag{19}$$

The following Eq. 20 estimates the cost equation to w_{ij} that represents the renewal of the weights:

$$\frac{\partial}{\partial w_{ij}} \sum_{c=1}^{C} \log_p (v^c) = \frac{\partial}{\partial w_{ij}} \sum_{c=1}^{C} \log \sum_g e^{-E(v^c,g)} - \log \sum_u \sum_g e^{-E(u,g)} \tag{20}$$

The first term calculates the average values of v_i^c, g_j when the visible level of RBM is leaded by the data (v^c), whereas the second term corresponds to the values of v_i, g_j when the data are "produced" by the model. An equivalent way of formulating, would suggest that every weight w_{ij} should change to become equal to Δw_{ij} (Eq. 21):

$$\Delta w_{ij} = \varepsilon_w \left(E_{data}\left[v_i h_j\right] - E_{model}\left[v_i h_j\right] \right) \tag{21}$$

The RBM model in this case starts approaching the actual values of the data. The first term $E_{data}\left[v_i h_j\right]$ can be calculated easily, since knowing the values of the units at the visible level, we can, by means of the above equations, calculate the reserved probability for each unit at the hidden level. Calculation of the second term, which presupposes the existence of samples from the model itself, is more difficult. In this paper we use optimization through Contrastive Divergence (CD) [27]. The steps that we used to procedure a single sample can be summarized as follows:

1. Take a training sample v, compute the probabilities of the hidden units and sample a hidden activation vector h from this probability distribution.
2. Compute the outer product of v and h and call this the *positive gradient*.
3. Use h, to sample a reconstruction v' of the visible units, and then resample the hidden activations h'. (Gibbs sampling step)
4. Compute the outer product of v' and h' and call this the *negative gradient*.
5. Let the update to the weight matrix W be the positive gradient minus the negative gradient, times some learning rate: $\Delta W = \varepsilon(uh^T - u'h'^{\mathrm{T}})$.
6. Update the biases a and b analogously: $\Delta a = \varepsilon(u - u'), \Delta b = \varepsilon(h - h')$.

The CD method gives lower energy to the actual data and much higher energy to the "reconstructions" resulting from them. This helps the model approach the actual data distribution.

Also, the Typhon employs the Multi-Task Grouping and Overlap Learning approach, combined with the optimal use of the OS-ELM and RBMs methodologies. The sliding windows (SLIW) [28] are used to partition the data stream. Using the SLIW technique, the system estimates a table of indices, after accepting the data flow vectors as input.

The data are divided in partitions of 1,000 samples with 400 of them overlapping between adjoining windows. This enables continuous and unintentional scanning of the data, which results in faster and more accurate control results. This happens because a small SLIW is much more likely to be more uniform than a larger one, and therefore it is more predictable. Each new sample is checked by the OS-ELM algorithm and force it in the appropriate RBM. Once the optimal model has been created, it is then applied to the control window of the SLIW. This process is followed until the input data of each window is completed. The total knowledge of the entire window is stored in MM1 (the first window). The MM1 is transferred in the window 2 as MLT, which means that the total knowledge of the first window is transferred to the second window. The process continues for all the data in window 2, and so on for all other windows, until the full analysis of the data flow is done. The full representation of the proposed process is presented in Fig. 2 below.

Fig. 2. The proposed Multi-Task Grouping and Overlap Learning approach

When the OS-ELM classification detects an attack type, the network flow data are directed to the corresponding RBM. If it is a false positive case, the data are directed to the next RBM and so on. If the output is a false alarm, then the traffic is redirected to the initial OS-ELM, which has the online learning potential and it re-examines the dataflow from the beginning as if it was a new one.

This architecture has been chosen due to the fact that in multi parametric and high complexity problems related to big data (like the one examined herein) the classification results are unstable especially regarding the analysis and the incorporation of the data flows.

The introduced architecture is a serious method of a hybrid combined resolution of the APT attacks. The system is flexible and effective and it not only offers a robust

approach but it also offers a faster convergence of the overall model. Finally, the proposed architecture can exploit the multi-tasking learning in order to enhance generalization. Each one of the developed RBMs is totally dedicated to the specific problem, which can result in bias and variance reduction. This can result in overfitting elimination and it offers a robust framework capable of coping with high complexity problems.

4 Datasets

Appropriate datasets were chosen that closely simulate Industrial Control System (ICS) communication and transaction data. They were used in the development and evaluation of the proposed model. Contained preprocessed network transaction data and preprocessed to strip lower layer transmission data, were used (e.g. TCP, MAC) [29]. Specifically, the Gas_Dataset that was chosen for the purpose of this research includes 26 independent parameters and 210,770 instances, from which 61,156 normal and 149,614 abnormal (7 categories of attacks: Naïve Malicious Response Injection (NMRI) 16,578, Complex Malicious Response Injection (CMRI) 15,466, Malicious State Command Injection (MSCI) 28,152, Malicious Parameter Command Injection (MPCI) 30,548, Malicious Equation Code Injection (MFCI) 20,628, Denial of Service (DoS) 11,022 and Reconnaissance (Recon) 27,220). The following Fig. 3 offers a graphical representation of the abnormal class distribution in the final Gas_Dataset.

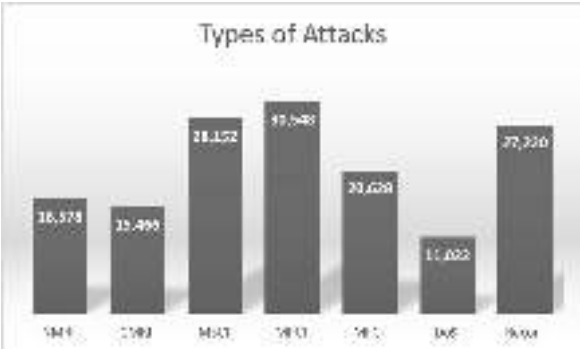

Fig. 3. Abnormal class distribution

The dataset is determined and normalized in the interval [−1, 1] in order to phase the problem of prevalence of features with wider range over the ones with a narrower range, without being more important. Also, the outliers and the extreme values spotted were removed based on the Inter Quartile Range technique [30]. The reader can find details regarding the dataset and the data collection and assessment methodology in [29].

4.1 Training the RBMs

The RBMs were trained based on an innovative application of the One Class Classification methodology. More specifically, according to this approach the system is exclusively trained with data related to a vulnerability of the network, in order to be able to identify the specific behavior – anomaly that has a potential relation with APT attacks. This is achieved by estimating the RBMs Reconstruction Error (RER), which is a basic criterion for the safe determination of the classes. It is calculated by using a threshold which is unique for each class and which emerged after several trial and error attempts, aiming in the optimal output of the system. If the error is higher than this predefined limit (which is characteristic for each class) then it is rejected as an unknown class and the data are redirected to the next RBM.

5 Results

In data cases using a machine learning classifier, for estimating the real error during training, the full probability density of both categories should be known. The classification performance is estimated by the development of a Confusion Matrix (CM), where the main diagonal values (top left corner to bottom right) correspond to correct classifications and the rest of the numbers correspond to very few cases that were misclassified. The following Table 1 presents the CM results of the OS-ELM:

Table 1. Confusion Matrix of the OS-ELM

Normal	NMRI	CMRI	MSCI	MPCI	MFCI	DoS	Recon
59,826	428	93	289	453	2	65	0
632	**15,944**	**0**	**2**	0	0	0	0
40	0	**15,426**	**0**	0	0	0	0
264	0	0	**27,888**	0	0	0	0
503	0	0	0	**29,900**	125	20	0
2	0	0	0	157	**20,469**	0	0
139	0	0	1	24	0	**10,858**	0
0	0	0	0	0	0	0	**2,220**

All of the performance metrics in this testing, were estimated based on the One Versus ALL approach, because it is a Multi-Class Classification case.

The numbers of misclassifications are related to the False Positive (FP) and False Negative (FN) indices appearing in the confusion Matrix. The True Positive rate (TPR) also known as Sensitivity the True Negative rate also known as Specificity (TNR) and the False Positive Rate (FPR) are defined by using Eqs. 22, 23, and 24 respectively [31].

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{22}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{23}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - TNR \tag{24}$$

The Precision (PRE) the Recall (REC), the F-Score and the Total Accuracy (TA) indices are defined as in Eqs. 25, 26, 27 and 28 respectively [31]:

$$\mathrm{PRE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{25}$$

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{26}$$

$$\mathrm{F-Score} = 2 \times \frac{\mathrm{PRE} \times \mathrm{REC}}{\mathrm{PRE} + \mathrm{REC}} \tag{27}$$

$$\mathrm{TA} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N}} \tag{28}$$

The following Table 2 present the analytical results of the proposed method.

Table 2. Classification Accuracy and Performance Metrics

Classifier	Fold	TA	RMSE	Precision	Recall	F-Score	AUC
OS-ELM	1st	98.51%	0.0548	0.980	0.980	0.9800	0.998
	2nd	98.63%	0.0541	0.990	0.990	0.9900	0.999
	3rd	97.96%	0.0482	0.976	0.976	0.9760	0.989
	4th	98.63%	0.0543	0.990	0.990	0.9900	0.996
	5th	98.98%	0.0578	0.989	0.989	0.9890	0.997
	6th	98.00%	0.0490	0.981	0.981	0.9810	0.995
	7th	98.60%	0.0549	0.986	0.986	0.9860	0.999
	8th	98.75%	0.0560	0.987	0.987	0.9870	0.999
	9th	98.28%	0.0567	0.986	0.986	0.9860	0.999
	10th	98.30%	0.0536	0.985	0.985	0.9850	0.999
	Avg	**98.46%**	**0.0539**	**0.985**	**0.985**	**0.985**	**0.997**

The 10-fold cross validation (10_FCV) is employed to obtain performance indices. Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

6 Discussion and Conclusions

This research introduces a highly innovative, reliable and effective anomaly detection system, which is based on advanced computational intelligence approaches [32–34]. The Cyber-Typhon performs a series of sophisticated anomaly detection equations, by using Multi-Task Learning and by effectively combining On-Line Sequential Extreme Learning Machines with Restricted Boltzmann Machines.

It ensures (in the most effective and intelligent way) the safe network communication among the interacting devices in a critical infrastructure environment. The proposed system, significantly enhances the security mechanisms of the Critical Infrastructures, which are a constant target due to their high importance. Also, this architecture offers the potential of developing a safe platform that can control and integrate network transactions. This can be done without the need for human intelligence or for a central authority. Also, it has been proven that collective intelligence technologies offer a smart solution for the determination of digital security and for monitoring of assets related to critical infrastructure.

The development of the model was based on the extremely effective OS-ELM algorithm, and on the multi-task learning method, which can perform transfer of knowledge between relative processes that are executed in parallel.

The training process was based on the employment of the RBMs that were trained based on the Unary Classification method. This was done by using specific datasets, each one of them corresponding to the behavior of a certain attack, in order to ensure the absolute reliability of the classifier.

The performance of the proposed system was tested on a multidimensional dataset of high complexity. This has resulted after an extensive research into the operation of critical infrastructure control systems and after comparisons, audits and tests. The target was the identification of the most appropriate limits, which express and realistically represent the classes they represent. The high accuracy of the results has greatly enhanced the validity of the general methodology.

It is important to mention that this particular model is presented for the first time in the literature. It constitutes an important guideline in further exploitation of intelligent technologies in the automations that compose industrial networks.

Proposals for the development and future improvements of this system, should focus on further optimizing the parameters of the RBMs used in order to achieve an even more efficient, accurate and quicker classification, capable of dividing even more precisely the boundaries between the situations of systems.

It would be important to study the equation-extension of the proposed algorithm with meta-learning methods. This could further improve the anomaly detection process. Finally, the introduced model can employ adaptive learning in order to gain self-improvement potentials. This would automate 100% the whole process.

References

1. Dedić, N., Stanier, C.: Towards differentiating business intelligence, big data, data analytics and knowledge discovery. In: Piazolo, F., Geist, V., Brehm, L., Schmidt, R. (eds.) ERP Future 2016. LNBIP, vol. 285, pp. 114–122. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58801-8_10
2. Kiran, M., Murphy, P., Monga, I., Dugan, J., Baveja, S.S.: Lambda architecture for cost-effective batch and speed big data processing. In: IEEE International Conference on Big Data (Big Data), Santa Clara, CA, pp. 2785–2792 (2015). https://doi.org/10.1109/bigdata.2015.7364082
3. Lin, J.: The Lambda and the Kappa. IEEE Internet Comput. **21**(5), 60–66 (2017). https://doi.org/10.1109/MIC.2017.3481351
4. Demertzis, K., Iliadis, L.: A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification. In: Sideridis, A.B., Kardasiadou, Z., Yialouris, C.P., Zorkadis, V. (eds.) e-Democracy 2013. CCIS, vol. 441, pp. 11–23. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11710-2_2
5. Krawczyk, B., Cano, A.: Online ensemble learning with abstaining classifiers for drifting and noisy data streams. Appl. Soft Comput. J. **68**, 677–692 (2018)
6. Baytas, I.M., Yan, M., Jain, A.K., Zhou, J.: Asynchronous multi-task learning. In: ICDM, pp. 11–20 (2016)
7. Zhang, Y.: Parallel multi-task learning. In: ICDM, pp. 629–638 (2015)
8. Chen, Q., Abdelwahed, S.: A model-based approach to self-protection in computing system. In: Proceeding CAC 2013 Proceedings of the ACM Cloud and Autonomic Computing Conference, Article no. 16 (2013)
9. Soupionis, Y., Ntalampiras, S., Giannopoulos, G.: Faults and cyber attacks detection in critical infrastructures. In: Panayiotou, C.G.G., Ellinas, G., Kyriakides, E., Polycarpou, M. M.M. (eds.) CRITIS 2014. LNCS, vol. 8985, pp. 283–289. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31664-2_29
10. Zhu, W.T., et al.: Detecting node replication attacks in wireless sensor networks: a survey. J. Netw. Comput. Appl. **35**(3), 1022–1034 (2012)
11. Cruz, T., et al.: Improving cyber-security awareness on industrial control systems: the CockpitCI approach. J. Inf. Warfare **13**(4) (2015). ISSN 1445 3347 (online), ISSN 445-3312 (printed)
12. Zhang, Y., Yeung, D.: A convex formulation for learning task relationships in multi-task learning. In: UAI, pp. 733–742 (2010)
13. Wang, J., Kolar, M., Srebro, N.: Distributed multi-task learning. In: AISTATS, pp. 751–760 (2016)
14. Xing, E.P., Ho, Q., Xie, P., Wei, D.: Strategies and principles of distributed machine learning on big data. Engineering **2**(2), 179–195 (2016)
15. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Linear algorithms for online multitask classification. In: COLT 2008, Helsinki, Finland, June 2008
16. Demertzis, K., Iliadis, L., Anezakis, V.: MOLESTRA: a multi-task learning approach for real-time big data analytics. In: 2018 Innovations in Intelligent Systems and Applications (INISTA), Thessaloniki, pp. 1–8 (2018). https://doi.org/10.1109/inista.2018.8466306
17. Demertzis, K., Iliadis, L., Spartalis, S.: A spiking one-class anomaly detection framework for cyber-security on industrial control systems. In: Boracchi, G., Iliadis, L., Jayne, C., Likas, A. (eds.) EANN 2017. CCIS, vol. 744, pp. 122–134. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65172-9_11

18. Demertzis, K., Iliadis, L.S., Anezakis, V.-D.: An innovative soft computing system for smart energy grids cybersecurity. Adv. Build. Energ. Res. **12**(1), 3–24 (2018). https://doi.org/10.1080/17512549.2017.1325401
19. Demertzis, K., Iliadis, L.: A computational intelligence system identifying cyber-attacks on smart energy grids. In: Daras, N.J., Rassias, T.M. (eds.) Modern Discrete Mathematics and Analysis. SOIA, vol. 131, pp. 97–116. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74325-7_5
20. Demertzis, K., Kikiras, P., Tziritas, N., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: network flow forensics using cybersecurity intelligence. Big Data Cogn. Comput. **2**, 35 (2018)
21. Demertzis, K., Tziritas, N., Kikiras, P., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: adaptive analytic lambda architecture for efficient defense against adversarial attacks. Big Data Cogn. Comput. **3**, 6 (2019)
22. Cyber-Security and Information Warfare. Cybercrime and Cybersecurity Research. NOVA Science Publishers. ISBN 978-1-53614-385-0. Chap. 5
23. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
24. El-Yaniv, R., Nisenson, M.: Optimal single-class classification strategies. In: Proceedings of the 2006 NIPS Conference, vol. 19, pp. 377–384. MIT Press (2007)
25. Munroe, D.T., Madden, M.G.: Multi-class and single-class classification approaches to vehicle model recognition from images. In: Proceedings of Artificial Intelligence and Cognitive Science, Portstewart (2005)
26. Zhang, N., Ding, S., Zhang, J., Xue, Y.: An overview on restricted Boltzmann machines. Neurocomputing **275**, 1186–1199 (2018). https://doi.org/10.1016/j.neucom.2017.09.065
27. Ma, X., Wang, X.: Convergence analysis of contrastive divergence algorithm based on gradient method with errors (2015). [Research article]
28. Dietterich, T.G.: Machine learning for sequential data: a review. In: Caelli, T., Amin, A., Duin, R.P.W., de Ridder, D., Kamel, M. (eds.) SSPR /SPR 2002. LNCS, vol. 2396, pp. 15–30. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-70659-3_2
29. Morris, T.H., Thornton, Z., Turnipseed, I.: Industrial control system simulation and data logging for intrusion detection system research. Int. J. Netw. Secur. (IJNS) **17**(2), 174–188 (2015)
30. Zwillinger, D., Kokoska, S.: CRC Standard Probability and Statistics Tables and Formulae, p. 18. CRC Press, Boca Raton (2000). ISBN 1-58488-059-7
31. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
32. Demertzis, K., Iliadis, L.: A bio-inspired hybrid artificial intelligence framework for cyber security. In: Daras, N.J., Rassias, M.Th. (eds.) Computation, Cryptography, and Network Security, pp. 161–193. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18275-9_7
33. Demertzis, K., Iliadis, L.: SAME: an intelligent anti-malware extension for Android ART virtual machine. In: Núñez, M., Nguyen, N.T., Camacho, D., Trawiński, B. (eds.) ICCCI 2015. LNCS (LNAI), vol. 9330, pp. 235–245. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24306-1_23
34. Demertzis, K., Iliadis, L.: Evolving smart URL filter in a zone-based policy firewall for detecting algorithmically generated malicious domains. In: Gammerman, A., Vovk, V., Papadopoulos, H. (eds.) SLDS 2015. LNCS (LNAI), vol. 9047, pp. 223–233. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17091-6_17