MOLESTRA: A Multi-Task Learning Approach for Real-Time Big Data Analytics

Konstantinos Demertzis[1], Lazaros Iliadis[2], Vardis-Dimitris Anezakis[3]
[1,2] School of Engineering, Department of Civil Engineering, Lab of Mathematics and Informatics (iSCE)
Democritus University of Thrace, University Campus, Kimmeria, Xanthi, Greece
[3]Department of Forestry and Management of the Environment and Natural Resources, Democritus University of Thrace, Orestiada, Greece
{kdemertz[1], danezaki[3]}@fmenr.duth.gr, liliadis@civil.duth.gr[2]

***Abstract* —**Modern critical infrastructures are characterized by a high degree of complexity, in terms of vulnerabilities, threats, and interdependencies that characterize them. The possible causes of a digital assault or occurrence of a digital attack are not simple to identify, as they may be due to a chain of seemingly insignificant incidents, the combination of which provokes the occurrence of scalar effects on multiple levels. Similarly, the digital explosion of technologies related to the critical infrastructure and the technical characteristics of their subsystems entails the continuous production of a huge amount of data from heterogeneous sources, requiring the adoption of intelligent techniques for critical analysis and optimal decision making. In many applications (e.g. network traffic monitoring) data is received at a high frequency over time. Thus, it is not possible to store all historical samples, which implies that they should be processed in real time and that it may not be possible to re-review old samples (one-pass constraint). We should consider the importance of protecting critical infrastructure, combined with the fact that many of these systems are cyber-attack targets, but they cannot easily be disconnected from their layout as this could lead to generalized operational problems. This research paper proposes a *Multi-Task Learning model* for *Real-Time & Large-Scale Data Analytics,* towards the Cyber protection of Critical Infrastructure. More specifically, it suggests the *Multi Overlap LEarning STReaming Analytics* (MOLESTRA) which is a standardization of the "*Kappa*" architecture. The aim is the analysis of large data sets where the tasks are executed in an overlapping manner. This is done to ensure the utilization of the cognitive or learning relationships among the data flows. The proposed architecture uses the *k-NN* Classifier with *Self Adjusting Memory* (k-NN SAM). MOLESTRA, provides a clear and effective way to separate the short-term from the long-term memory. In this way the temporal intervals between the transfer of knowledge from one memory to the other and vice versa are differentiated.

Keywords—"Kappa" Architecture, Multi-Task Learning, Big Data, Data Streams, Critical Infrastructure Protection, Advanced Persistent Threat

I. INTRODUCTION

A. Cybersecurity Protection of Critical Infrastructures

Protection of a country's *Critical Infrastructure* is a very important task, affecting prosperity and having serious consequences in political and strategic level. This infrastructure is directly related to physical, social, economic or environmental prosperity. Although this is a timeless requirement, the protection of critical infrastructure nowadays has a strong correlation with the security of information and telecommunication technologies and systems. Automation, remote control and internet connection are the most up-to-date methods by which networks of these infrastructures can improve productivity and quality of their services [1].

In this respect, the secure and efficient management of industrial IT systems and sophisticated automation ones such as *Industrial Control Systems* (ICS) and SCADA, is of paramount importance. These systems have been integrated into *Critical Infrastructure Networks* with the aim of successfully completing specialized activities with precision and efficiency [2]. Intelligent identification- classification of possible attacks against these systems, is a very important and overriding work. Vulnerabilities related to operational control may lead to the partial or total collapse of the critical infrastructure network with incalculable social and economic consequences.

B. Large-Scale Data Analytics

The magnitude of the data generated by the above-mentioned IT systems (ICS and SCADA) has led to an exponential increase of the information volume. The efficient management of the obtained information, and the huge storage requirements, demand the upgrade of conventional techniques, as they were not designed to process such a large volume of data. The *Velocity* of the data collection process and the need for "real-time" processing, combined with the variety of unstructured or semi-structured forms of data, creates very serious organizational and storage problems. Their *Variability*, which differs from their *Variety* and their importance over time, leads to many challenges and difficulties related to their proper exploitation. It is therefore extremely important to develop and to use management systems (MASY) for large Volumes of data. MASY can provide proper tools to consider and analyze information obtained by industrial information systems (IIS) embedded in the *Critical Infrastructure Networks* (CIN). This can lead to timely, valid and documented strategic decisions. Moreover, these systems should provide intelligent methods for the discovery of secret knowledge, hidden in the *Visualization* data. The *Veracity* of this knowledge and the determination of its real *Value* should also be accessed [3].

C. Real Time Big Data Stream Processing

The architectures of Big Data systems standardization methods have been put on a new basis, taking into account the evolution of the needs for managing and analyzing large amount of data generated by systems such as industrial IT automation, coupled with real-time data processing requirements [4]. These architectures should be capable to embed *Real Time Data Stream* (RTDS) processing algorithms (PRAL), capable to dynamically adjust in new prototypes or data. It is a fact that the data generated by industrial automation systems (IAS) used in CIN are provided in a staggered and sequential order. Therefore, the error should be calculated at each iteration and the purpose of the

978-1-5386-5150-6/18/$31.00 ©2018 IEEE

PRAL should be the minimization of the overall cumulative error by employing the following function 1 [5, 6]:

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

where $x_j \in R^d, w \in R^d$ and $y_j \in R$. We suppose that X$i \times d$ is the data table and Y$i \times 1$ is the table of target values as it emerges after the arrival of the first i data points. Supposing that the covariance table $\Sigma_i = X^T X$ is reversable, the optimal solution $f^*(x) = \langle w^*, x\rangle$ for the linear least squares problem is given by the equation 2 below [6]:

$$w^* = (X^T X)^{-1} X^T Y = \Sigma_i^{-1} \sum_{j=1}^{i} x_j y_j \quad (2)$$

After estimating the covariance matrix $\Sigma_i = \sum_{j=1}^{i} x_j x_j^T$ and considering that the time complexity (TICO) is $O(id^2)$, the reverse of the table $d \times d$ leads to a TICO equal to $O(d^3)$. The rest of the multiplication has a TICO of $O(d^2)$ order and the total temporal complexity is $O(id^2 + d^3)$ *where* $i = 1,2,\dots,n$. It should be clarified that it is necessary to recalculate the solution after the arrival of each new data point. A naïve approach could have TICO equal to $O(n^2 d^2 + nd^3)$, which would be catastrophic for its application in demanding environments characterized by rapid changes like the one examined herein [6].

D. Multi-Task Learning

Multi-Task Learning (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 [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 [8].

The following approaches are characteristic cases of MTL [8]:

- *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.

II. METHODOLOGY-THEORETICAL BACKGROUND

A. Justification of the "Kappa" Architecture employment

In the cases of batch data architectures, data is treated as static, assuming all data is available during training. These architectures produce a final model, using all the available data that are being examined simultaneously. Instead, in cases of knowledge extraction and model implementation for real-time decision making, ML architectural implementations are required capable to learn from a series of data provided over time. At the same time they will be able to keep up-to-date whenever any new batch of data arrives [9, 10]. A typical architecture suitable for many of the Big Data-related challenges which employs ML algorithms to reveal hidden knowledge from streams, is the so-called "Kappa" architecture [11], presented in the figure below.

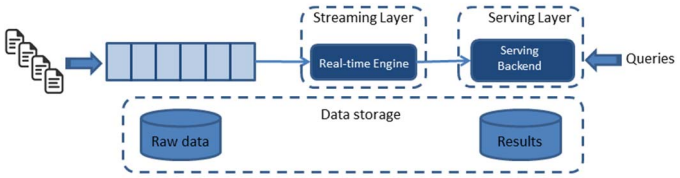

Fig 1. Kappa architecture (www.oreilly.com)

This architecture includes a real time system known as real time engine (RETE) which is used to analyze batches of data. For each new sample, a small gradual updating of the model takes place, which gradually improves as more data flows. This approach is based on *online* or *incremental learning,* to solve real-time data flow analysis problems [11].

The comparatively small storage capacity and resources requirements of this architecture is its main advantage. This is due to the fact that after the model's update using the new batches of data, the system does not need to store heavy data load in memory and thus its hardware requirements are reduced. This architecture allows fast models' updates with low computational complexity, and for this reason it is considered as the most suitable for the cases with strict time limitations like to one examined herein (Digital Security of CIN) where the response time must be milliseconds. Finally, the flexible scheme of this approach is adjusted more efficiently in the changes of data caused by a noisy field which escalates in time. Most of the learning approaches used to analyze data flows are characterized by a "*memory clearance*" process, based on which the learning methods "*forget*" the past. In this way, the significance of the past data is gradually reduced and the model is adjusted to the new conditions of its environment. This ensures the safe function of the model, even under the most sudden changes that might emerge, (e.g. due to noise in the input data, outliers or extreme values). However, this characteristic is connected to a disadvantage of the data flow analysis, known as "*Catastrophic Interference*" or "*Catastrophic Forgetting*" which is related to the fact that the algorithm completely forgets the information learned during the training phase, in cases of memory connection models [11].

B. Multi-Task Grouping and Overlap Learning

It is well known that data flow processing methods and corresponding algorithms, address serious challenges associated with large data volume handling procedures and their

effectiveness. In most cases, good preparation and methodological determination of their operating parameters is needed to avoid undesirable variations in accuracy (associated with frequent model updates) and instability or loss of generalization, which may be due to corrupted and noisy data. The aim is to create a precise and consistent classification model, that is, to remain stable with the presence of new information without depriving of the generalization potential, eliminating the "Catastrophic Interference" phenomenon. The proposed model employs a *multi-task grouping and overlap ML* approach, with an innovative special version of the "*Kappa*" architecture. It employs the k-NN-SAM algorithm, in order to analyze and access in real time the flows of data originating from Smart Grids. The final aim is the classification and detection of undesirable situations of digital security, related to Advanced Persistent Threat (APT) attacks [12].

The reason for using the multi-task learning technique in this research is the fact that very often in multi-factorial problems of high complexity such as the one under consideration, the results of the forecast are multi-variable (MUVA). The current MUVA can be attributed to the sensitivity of correlation models, the heterogeneity of data flows, and the mathematical functions used to describe a complex relationship, as in this case.

The most important advantage of the *multi-task learning* techniques, is the fact that it can lead to very stable forecasting models, offering generalization of the system under new conditions with impartiality, which is a key issue in the field of ML. Also, in general a multi-task learning method offers better prediction, as the overall behavior of a multiple model is less noisy than a corresponding single one. Still, these models can reduce bias, they can reduce variance and eliminate overfitting, delivering robust models capable of responding to high complexity problems [7, 8].

C. *k-NN SAM*

The *k-NN SAM* algorithm is inspired by the research field of the human memory and more specifically by the short term/long term memory model (STM & LTM) [13]. The information that reach STM through sensors are accompanied by relevant knowledge coming from the LTM. Information that receives a lot of attention and is deemed important is transferred to LTM in the form of Synaptic Consolidation. The capacity of the STM is quite limited and the information is kept for a very short time, unlike the LTM, which can maintain information for several years. A typical example of how human memory works in this field is the fact that we never forget the way we do bicycle, no matter how many years have passed since our last bicycle ride.

The k-NN SAM architecture, is partly inspired by this model, presenting analogies such as the self-explanatory separation of short-term and long-term memory, different time intervals between memories and the transfer of knowledge from STM to LTM and vice versa. The implementation of this algorithm as a data stream categorization model is based on the general assumption that new data is more relevant to current forecasts, but prior knowledge is required to properly rank them. The optimal combination of the two processing levels can minimize errors and increase classification precision [13]. The k-NN SAM approach is presented in the following figure 2 below.

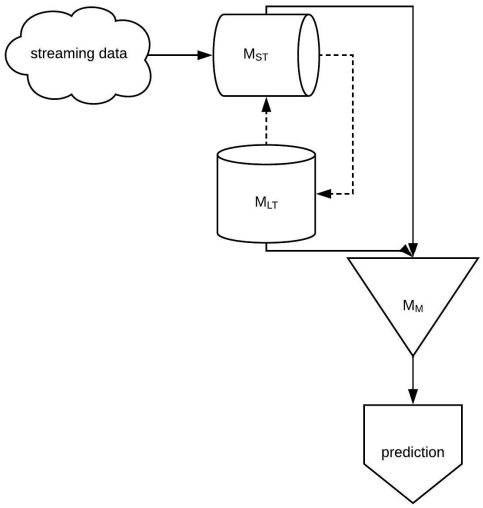

Fig 2. k-NN SAM architecture

The memory comprises of the Short-Term (M_{ST}), the Long-Term (M_{LT}) and the Merged one (M_M). Each memory is a subset of the $R^n \times \{1, \dots, c\}$ that is characterized by variable length that is increased or reduced during the adjustment period. The M_{ST} represents the current idea. It is a dynamic sliding window containing the most recent examples m of the data flow [13]:

$$M_{ST} = \{(x_i, y_i) \in R^n \times \{1, \dots, c\} \,|i = t - m + 1, \dots, t\} \quad (3)$$

The M_{LT} retains all the old information, which do not contradict to the ones of M_{ST}. The M_{LT} is not a continuous part of the data flow but it is a set of points p:

$$M_{LT} = \{(x_i, y_i) \in R^n \times \{1, \dots, c\} \,|i = 1, \dots, p\} \quad (4)$$

The unification of both memories is the M_M memory:

$$M_M = M_{ST} \cup M_{LT} \quad (5)$$

Every set includes the weighted k-NN classifier:

$$R^n \times \{1, \dots, c\}, kNN_{M_{ST}}, kNN_{M_{LT}}, kNN_{M_M} \quad (6)$$

The k-NN function assigns a label to a given data point x based on the set $Z = \{(x_i, y_i) \in R^n \times \{1, \dots, c\} \,|i = 1, \dots, n\}$:

$$kNN_Z(x) = argmax \left\{ \sum_{x_i \in N_k(x,Z) | y_i = \hat{c}} \frac{1}{d(x_i, x)} | \hat{c} = 1, .., c \right\} (7)$$

where $d(x_i, x)$ is the Euclidean distance between two points and the $N_k(x, Z)$ returns the set of the k nearest neighbors of x in Z [13].

III. PROPOSED APPROACH

A. *The proposed MOLESTRA*

The MOLESTRA employs the Multi-Task Grouping and Overlap Learning approach, combined with the optimal use of the k-NN SAM methodology. The sliding windows (SLIW) are used to partition the data stream. Using the SLIW technique [14], the system estimates a table of indices, after accepting the data flow vectors as input. The data are divided in partitions of 1000

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 k-NN SAM algorithm and it is assessed by using the *objective function kappa* [15]. The kappa coefficient is a powerful statistical measure for evaluating the performance of the algorithm. The knowledge created in each sample is stored in the M_{ST}, and the accumulated knowledge from the sample examination process is stored in the M_{LT}. 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, which is the combination (conjunction) of M_{ST} and M_{LT}, is stored in M_{M1} (the first window). The M_{M1} is transferred in the window 2 as M_{LT}, 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 figure 3 below.

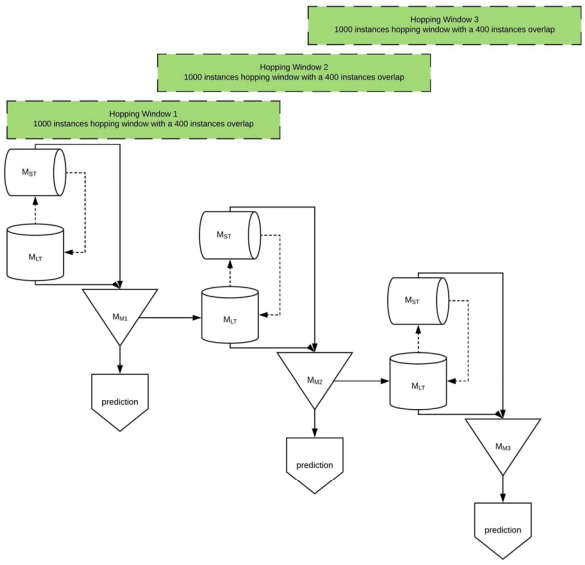

Fig 3. MOLESTRA architecture

Various experiments were performed, by changing either the size of the sliding window or the degree of overlapping, or alternatively by changing both.

IV. Literature Review

The MOLESTRA is a multi-task intelligence computer security technique [16, 17, 18, 19, 20, 21, 22, 23, 24]. In general, existing MTL methods can be categorized into two main categories: learning with feature covariance [25] and learning with task relations [26]. Different from prior solutions to distributed MTL, which are focused on the former category [27], our proposed DMTRL falls into the latter category. On the other hand, distributed machine learning has attracted more and more interests recently [28]. 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 adversarially chooses task to learn. Cavallanti et al. [29] exploited online MTL with a given task relationship encoded in a matrix, which is known beforehand. Parallel multi-task learning aims to develop parallel computing algorithms for MTL in a shared-memory computing environment. Recently, Zhang [30] proposed a parallel MTL 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. [31] proposed a distributed algorithm for MTL by assuming that different tasks are related through shared sparsity. In another work [32], asynchronous distributed MTL method is proposed for MTL with shared subspace learning or shared feature subset learning. Different from the above-mentioned approaches, our method aims at solving MTL by learning task relationships from data, which can be positive, negative, or unrelated, via a task-covariance matrix.

V. DATASETS

A. Idea and Features Extraction

In order to implement the research and to evaluate the proposed model, an appropriate set of data was chosen that most closely simulates the communication and the transaction data between ICS. Network traffic analysis, and feature extraction methodology, were determined based on the functionality of ICS, namely on the verification of the reliable and error-free data upload and capture mechanisms. The sets also include data logs from flagged network transactions during the regular operation of certain ICSs, as well as transactions during 35 different cyber-attacks. Finally, they also include measurements of physiological behavior as well as abnormalities detected during attacks, which were simulated in a virtual ICS environment. It should be noted that the configured virtual systems have no physical limits to the size of the simulation, therefore the virtual platform on which the data was collected was formed by scaling the ICS. This was done in order to represent their operational situations in the most realistic way.

B. Data

Appropriate datasets were chosen that closely simulate 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) namely [33]:

- The water_tower_dataset includes 23 independent parameters and 236,179 instances, from which 172,415 normal and 63,764 outliers. Totally 86,315 normal instances were used in the training phase (water_train_dataset) whereas the rest 86,100 normal instances and 63,764 outliers comprised the water_test_dataset.

- The gas_dataset includes 26 independent features and 97,019 instances, from which 61,156 normal and 35,863 outliers. The training of the algorithm was done with the gas_train_dataset that contains 30,499 normal instances, whereas the rest 30,657 normal instances and 35,863 outliers, belong to the gas_test_dataset.
- Finally, the electric_dataset includes 128 independent variables with 146,519 instances, from which 90,856 normal and 55,663 outliers. The training was performed based on the electric_train_dataset comprising of 45,402 normal instances, whereas the rest 45,454 normal and the 55,663 outliers, belong to the electric_test_dataset.

These sets contain data logs from a Gas Pipeline, a Lab Scale Water Tower and a Lab Scale Electric Transmission system. The logs include flagged network transactions during the normal operation of specific ICSs, as well as transactions during 35 different cyber-attacks. In addition to the logs, our data include normal behavior measurements, as well as abnormalities detected during attacks that were simulated in a virtual ICS environment. Details regarding the dataset, their choice and assessment can be found in [33].

The dataset is determined and normalized to 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 [33]. Also, the outliers and the extreme values spotted were removed based on the Inter Quartile Range technique [34]. The reader can find details regarding the dataset and the data collection and assessment methodology in [33].

VI. RESULTS

The data flow classifiers aim to capture the actual error. Comparison of classification performance is done in terms of *Accuracy Kappa* statistic and *Kappa-Temporal* statistic, using the traditional immediate setting. The true label is presented right after the instance has been used for testing, or after the delayed setting, where there is a delay between the time an instance is presented and its true label becomes available [15].

The Kappa statistic is a popular measure for benchmarking classification accuracy under class imbalance and is used in static classification scenarios as well as streaming data classification. The Kappa statistic or Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories. The definition of κ is [15]:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e} \quad (11)$$

where p_o is the relative observed agreement among raters (identical to accuracy), and p_e is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly seeing each category. If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters other than what would be expected by chance (as given by p_e), $\kappa \approx 0$. Considering the presence of temporal dependencies in data streams, the Kappa-Temporal statistic is defined as follows [15]:

$$\kappa_T = \frac{p - p_{per}}{1 - p_{per}} \quad (12)$$

where p_{per} is the accuracy of the Persistent classifier.

The Kappa-Temporal statistic may take values from 1 down to $-\infty$. The interpretation is similar to that of κ. If the classifier is perfectly correct then $\kappa_{per} = 1$. If the classifier is achieving the same accuracy as the Persistent classifier, then $\kappa_{per} = 0$. Classifiers that outperform the Persistent classifier fall between 0 and 1. Sometimes it may happen that $\kappa_{per} < 0$, which means that the reference classifier is performing worse than the Persistent classifier baseline.

Therefore, using κ_{per} instead of κ, we will be able to detect misleading classifier performance for data that has temporal dependence. For highly imbalanced, but independently distributed data, the majority class classifier may beat the Persistent classifier and thus using κ_{per} will not be sufficient. Overall, κ_{per} and κ measures can be seen as orthogonal, since they measure different aspects of performance. Hence, for a thorough evaluation we recommend measuring and combining both.

In this study, the implementation of the system was done with learning evaluation instances using 10,000 records and the validation of the results was done by the *Prequential Evaluation* method [35] in which each case is first used to test the model and then it is trained. Totally, 1000 instances were used for the training window. It should be noted that the results presented in the table below are the total average for each metric performance and for the whole process. The following tables 1, 2 and 3 presents indicative results from the first five (5) windows and the final results of the propose method from the three datasets.

Table 1. *Results from water_tower_dataset*

Window	Classification Accuracy & Performance Metrics (**water_tower_dataset**)		
	Accuracy	Kappa Statistic	Kappa Temporal Statistic
Window 1	88.18%	70.31%	71.28%
Window 2	88.52%	70.83%	71.67%
Window 3	89.16%	71.08%	71.81%
Window 4	89.58%	71.44%	72.01%
Window 5	89.95%	71.80%	72.46%
Average of all windows	**92.86%**	**73.11%**	**74.03%**

Table 2. *Results from gas_dataset*

Window	Classification Accuracy & Performance Metrics (**gas_dataset**)		
	Accuracy	Kappa Statistic	Kappa Temporal Statistic
Window 1	90.57%	72.56%	74.37%
Window 2	90.91%	72.94%	74.72%
Window 3	91.22%	73.10%	74.96%
Window 4	91.63%	73.58%	74.88%
Window 5	92.04%	74.13%	75.29%
Average of all windows	**93.27%**	**74.26%**	**75.84%**

Table 3. *Results from electric_dataset*

Window	Classification Accuracy & Performance Metrics (**electric_dataset**)		
	Accuracy	Kappa Statistic	Kappa Temporal Statistic
Window 1	87.32%	70.03%	71.18%
Window 2	88.18%	70.96%	71.69%
Window 3	88.20%	70.98%	71.83%
Window 4	88.96%	71.33%	72.31%

Window 5	89.19%	71.99%	72.93%
Average of all windows	**91.54%**	**72.53%**	**73.15%**

The following table 4 below explains the meaning of the Kappa range values which determine the credibility of observers and markers.

Table 4. Kappa Reliability

Kappa	Reliability
0.00	no reliability
0.1 – 0.2	minimum
0.21 – 0.40	little
0.41 – 0.60	moderate
0.61 – 0.80	important
≥ 0.81	maximum

As we can see, there is considerable reliability in all cases tested, which also strengthens the overall reliability and usability of the proposed model.

VII. CONCLUSIONS - DISCUSSION

A. Elements of Innoviation

An important innovation of MOLESTRA is the use of Multi-task learning techniques capable to solve a multi-dimensional and complex IT security problem. Multi-task learning systems simulate in a realistic way the functioning of biological learning, the experiential mode of human memory, memories and more generally, the ways in which the brain models the spatio temporal experiences. Also, an important innovation is the separation of short-term from long-term memory and the transfer of knowledge between them as a learning system. Essentially, this technique greatly enhances the way in which the learning and knowledge extraction systems work, as it creates the prospect of creating heterogeneous systems to which learning transfer can be applied. Finally, it should not be overlooked that an equally important innovation is the fact of adding artificial intelligence to the level of real-time analysis of industrial equipment. This fact greatly enhances the active defense mechanisms of critical infrastructures. This innovation provides important solutions and improves the way IT systems work and respond to cyber-attacks.

B. Comments on the results

Looking at the results, it is important to note that the use of the proposed MOLESTRA is a reliable and perhaps “optimal” practice in solving a real IT security problem. Essentially this practice is to categorize data streams more accurately, using a special form of knowledge transfer between parallel tasks, while ensuring generalization and reliability.

Also, with this technique the model is led to a relatively smooth but great rhythm of learning, which determines how quickly learning is converging. A high learning rate can lead to faster convergence and oscillation around optimal weight values, while the low rate of learning results in slower convergence and can lead to trapping at local extremes. The high rate of convocation of learning is confirmed by the high accuracy rates achieved by the model (92.86%), considering that data flows are much smaller than a batch data set that has to be evaluated. The continuous "improvement of forecasting" by this technique is interpreted as a result of knowledge transfer, which also reflects the quality of the model, with its continuous upward adjustment to continuously increased classification precision rates.

The high learning precision of the algorithm and the mild mutation of the multi-task learning method, are due to the successful identification of local extremes (which may be included in data streams or in sliding windows) as new areas of the multidimensional solution space are explored. On the contrary, if the rate of "mutation" was too high, it could lead to a reduction in the exploitation of highly suitable areas of the solution space, and it could trap the system into solutions that do not generalize. [15, 35].

The Kappa coefficient links the level of observed agreement to the level of random agreement, and it estimates the variability in each observer-rater variation that occurs when the same observer evaluates differently in repeated evaluations of the same size. The maximum value of Kappa index represents the complete agreement and thus the maximum reliability between observers – markers. The minimum value 0 of the index is interpreted as there is only a random agreement between the observer - markers and thus no reliability between them [15, 35].

C. Summarizing

Summarizing we can say that MOLESTRA is an innovative approach of analysis and classification of large scale data flows, that uses the k-NN SAM algorithm in order to identify deviations of the ICS, which in most of the cases are due to cyberattacks. MOLESTRA was based in the multi-task learning approach, where knowledge is transferred between relative tasks executed in parallel. The implementation was based on the k-NN SAM algorithm, which provides knowledge transfer potential. Based on this, data flows were tested with a *Multi-Task Grouping and Overlap Learning* technique with sliding learning windows. In this way, effective and real time identification of cyber-attacks was achieved. This approach, and moreover the philosophy of active security, significantly enhances the Critical Infrastructure control. Three multidimensional datasets were used to test the efficiency of the proposed system. These datasets emerged after an extensive research on the operation of the ICS (SCADA, DCS, PLC). They realistically state the operating conditions of these devices in normal conditions and in situations where they are subject to cyber-attacks. The very high precision results that have emerged greatly enhance the general methodology followed, although the degree of difficulty and realism that has been added has created highly multifaceted issues of thorough investigation and reflection.

MOLESTRA is a robust learning system operating in uncertain environments, that offers a very reliable solution in the multifactorial and complicated problem of information systems security, like the classification of network traffic. It is also important to stress that system uncertainty as well as the instability of learning algorithms support the adoption of the multi-task method that normalizes the noisy field and yields consistent results that are capable of evaluating the problem.

Finally, it is important to mention the contribution of the *k-NN SAM* algorithm and the high success rates, due to the fact that this problem is located in a high-dimensional space where this

algorithm is more efficient. Finally, the optimal combination of the short-term and the long-term process levels has proven to minimize the errors and to increase the classification accuracy.

D. Future Work

Proposals for the development and future improvements of this system, should focus on further optimizing the parameters of the *k-NN SAM* algorithm used in MOLESTRA, so that an even more efficient, accurate, and faster classification process is achieved. Also, we should consider ways for further optimizing the use of the *sliding window*, so that maximum knowledge transfer is achieved in the best possible way, utilizing the short and long-term memory mechanisms. Finally, an additional element (that could be studied in the direction of future expansion) is the operation of MOLESTRA with methods of self-improvement and redefinition of its parameters. This could be done by using adaptive learning, so that we can automate the process of cyberattacks identification.

REFERENCES

[1] W. Hurst, M. Merabti , Fergus P. (2014) A Survey of Critical Infrastructure Security. In: Butts J., Shenoi S. (eds) Critical Infrastructure Protection VIII. ICCIP 2014. IFIP Advances in Information and Communication Technology, vol 441. Springer, Berlin, Heidelberg.

[2] C. Wang, L. Fang and Y. Dai, (2010), A simulation environment for SCADA security analysis and assessment, Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, vol. 1, pp. 342–347.

[3] Dedić, N.; Stanier, C. (2017), Towards Differentiating Business Intelligence, Big Data, Data Analytics and Knowledge Discovery, 285. Berlin ; Heidelberg: Springer International Publishing. ISSN 1865-1356. OCLC 909580101.

[4] M. Kiran, P. Murphy, I. Monga, J. Dugan and S. S. Baveja, (2015), Lambda architecture for cost-effective batch and speed big data processing, IEEE International Conference on Big Data (Big Data), Santa Clara, CA, pp. 2785-2792, doi: 10.1109/BigData.2015.7364082.

[5] Aretz, Kevin; Bartram, Söhnke M.; Pope, Peter F. (2011), Asymmetric Loss Functions and the Rationality of Expected Stock Returns, International Journal of Forecasting. 27 (2): 413–437. doi:10.1016/j.ijforecast.2009.10.008. SSRN 889323.

[6] Harold J. Kushner and G. George Yin, (2003), Stochastic Approximation Algorithms and Applications, New York: Springer-Verlag, 1997. ISBN 0-387-94916-X; 2nd ed., titled Stochastic Approximation and Recursive Algorithms and Applications, ISBN 0-387-00894-2.

[7] Liu, Sulin, Sinno Jialin Pan and Qirong Ho, (2017), Distributed Multi-Task Relationship Learning, KDD '17 Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 937-946, doi>10.1145/3097983.3098136.

[8] Bernardino Romera Paredes, Andreas Argyriou, Nadia Berthouze, Massimiliano Pontil, (2012), Exploiting Unrelated Tasks in Multi-Task Learning, Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR 22:951-959.

[9] M. Kiran, P. Murphy, I. Monga, J. Dugan and S. S. Baveja, (2015), Lambda architecture for cost-effective batch and speed big data processing, IEEE International Conference on Big Data (Big Data), Santa Clara, CA, 2015, pp. 2785-2792, doi: 10.1109/BigData.2015.7364082

[10] Y. Yamato, H. Kumazaki and Y. Fukumoto, (2016), Proposal of Lambda Architecture Adoption for Real Time Predictive Maintenance, Fourth International Symposium on Computing and Networking (CANDAR), Hiroshima, pp. 713-715, doi: 10.1109/CANDAR.2016.0130

[11] J. Lin, (2017), The Lambda and the Kappa, in IEEE Internet Computing, vol. 21, no. 5, pp. 60-66, doi: 10.1109/MIC.2017.3481351.

[12] I. Jeun , Y. Lee , Won D., (2012), A Practical Study on Advanced Persistent Threats. In: Kim T. et al. (eds) Computer Applications for Security, Control and System Engineering. Communications in Computer and Information Science, vol 339. Springer, Berlin, Heidelberg.

[13] V. Losing, B. Hammer and H. Wersing, (2016), KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift, IEEE 16th International Conference on Data Mining (ICDM), Barcelona, 2016, pp. 291-300, doi: 10.1109/ICDM.2016.0040.

[14] Dietterich T.G. (2002) Machine Learning for Sequential Data: A Review. In: Caelli T., Amin A., Duin R.P.W., de Ridder D., Kamel M. (eds) Structural, Syntactic, and Statistical Pattern Recognition. SSPR /SPR 2002. Lecture Notes in Computer Science, vol 2396. Springer.

[15] Žliobaitė, I., Bifet, A., Read, J. et al, (2015), Evaluation methods and decision theory for classification of streaming data with temporal dependence, Mach Learn 98: 455. https://doi.org/10.1007/s10994-014-5441-4.

[16] Demertzis K., Iliadis L. (2014). A Hybrid Network Anomaly and Intrusion Detection Approach Based on Evolving Spiking Neural Network Classification. In: Sideridis A., Kardasiadou Z., Yialouris C., Zorkadis V. (eds) E-Democracy, Security, Privacy and Trust in a Digital World. e-Democracy 2013. Communications in Computer and Information Science, vol 441. Springer, Cham

[17] Demertzis K., Iliadis L. (2014). Evolving Computational Intelligence System for Malware Detection, In: Advanced Information Systems Engineering Workshops, Lecture Notes in Business Information Processing, 178, 322-334. doi: 10.1007/978-3-319-07869-4_30

[18] Demertzis K., Iliadis L. (2014, April). Bio-Inspired Hybrid Artificial Intelligence Framework for Cyber Security. In: Daras N., Rassias M. (eds) Computation, Cryptography, and Network Security. Springer, Cham

[19] Demertzis K., Iliadis L. (2014, November). Bio-Inspired Hybrid Intelligent Method for Detecting Android Malware, In: Iliadis L., Papazoglou M., Pohl K. (eds) Advanced Information Systems Engineering Workshops. CAiSE 2014. Lecture Notes in Business Information Processing, vol 178. Springer, Cham

[20] Demertzis K., Iliadis L. (2015, April). Evolving Smart URL Filter in a Zone-based Policy Firewall for Detecting Algorithmically Generated Malicious Domains. In: Gammerman A., Vovk V., Papadopoulos H. (eds) Statistical Learning and Data Sciences. SLDS 2015. Lecture Notes in Computer Science, vol 9047. Springer, Cham.

[21] Demertzis K., Iliadis L. (2015, September). SAME: An Intelligent Anti-Malware Extension for Android ART Virtual Machine. In: Núñez M., Nguyen N., Camacho D., Trawiński B. (eds) Computational Collective Intelligence. Lecture Notes in Computer Science, vol 9330. Springer.

[22] Demertzis K., Iliadis L. (2016), Computational Intelligence Anti-Malware Framework for Android OS, Vietnam J Comput Sci (2017) 4: 245. https://doi.org/10.1007/s40595-017-0095-3.

[23] Demertzis K., Iliadis L. (2016), Ladon: A Cyber-Threat Bio-Inspired Intelligence Management System, Journal of Applied Mathematics & Bioinformatics, vol.6, no.3, 2016, 45-64, ISSN: 1792-6602 (print), 1792-6939 (online), Scienpress Ltd, 2016.

[24] Demertzis K., L. S. Iliadis, V.-D. Anezakis, An innovative soft computing system for smart energy grids cybersecurity, Advances in Building Energy Research, pp. 1-22, Taylor & Francis.

[25] G. Obozinski, B. Taskar, and M. I. Jordan, (2010), Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing, 20(2):231–252.

[26] Y. Zhang and D. Yeung, (2010), A convex formulation for learning task relationships in multi-task learning. In UAI, pages 733–442.

[27] J. Wang, M. Kolar, and N. Srebro, (2016), Distributed multi-task learning with shared representation. arXiv preprint arXiv:1603.02185.

[28] M. Balcan, A. Blum, S. Fine, and Y. Mansour, (2012), Distributed learning, communication complexity and privacy, In COLT, p 26.1–26.22.

[29] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, (2010), Linear algorithms for online multitask classification. JMLR, 11:2901–2934.

[30] Y. Zhang, (2015), Parallel multi-task learning. In ICDM, pages 629–638.

[31] J. Wang, M. Kolar, and N. Srebro, (2016), Distributed multi-task learning. In AISTATS, pages 751–760.

[32] I. M. Baytas, M. Yan, A. K. Jain, and J. Zhou, (2016), Asynchronous multi-task learning. In ICDM, pages 11–20.

[33] Thomas H. Morris, Zach Thornton, Ian Turnipseed, (2015), Industrial Control System Simulation and Data Logging for Intrusion Detection System Research, International Journal of Network Security (IJNS), Vol.17, No.2, PP.174-188.

[34] Zwillinger, D., Kokoska, S., (2000), CRC Standard Probability and Statistics Tables and Formulae, CRC Press. ISBN 1-58488-059-7 page 18.

[35] João Vinagre, Alípio Mário Jorge, João Gama, (2014), Evaluation of recommender systems in streaming environments, Workshop on 'Recommender Systems Evaluation: Dimensions and Design' (REDD 2014), held in conjunction with RecSys 2014. October 10, 2014, Silicon Valley, United States; doi:10.13140/2.1.4381.5367.