D3.4 Machine Learning empowered intrusion detection using Honeypots' data v1

WP3 – Cyber security risk assessment & Beyond – Sphinx Intelligence

Version: 1.00

SPHINX

A Universal Cyber Security Toolkit for Health-Care Industry

Disclaimer

Copyright message

Document information

Grant Agreement Number	826183	Acronym	SPHINX		
Full Title	A Universal Cyber Security Toolkit for Health-Care Industry				
Topic	SU-TDS-02-2018 Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures				
Funding scheme	RIA - Research and Innovation action				
Start Date	1st January 2019	**Duration**	36 months		
Project URL	http://sphinx-project.eu/				
EU Project Officer	Reza RAZAVI (CNECT/H/03)				
Project Coordinator	Dimitris Askounis, National Technical University of Athens - NTUA				
Deliverable	D3.4. Machine Learning empowered intrusion detection using Honeypots' data v1				
Work Package	WP3 – Cyber security risk assessment & Beyond – Sphinx Intelligence				
Date of Delivery	**Contractual**	M18	**Actual**	18	
Nature	R - Report	**Dissemination Level**	P - Public		
Lead Beneficiary	AiDEAS				
Responsible Author	Dr Serafeim Moustakidis	**Email**	s.moustakidis@aideas.eu		
		Phone			
Reviewer(s):	Yannis Nikoloudakis (HMU), Dimitris Apostolakis (FINT)				
Keywords	Machine learning, deep learning, intrusion detection				

Document History

Version	Issue Date	Stage	Changes	Contributor
0.10	24/01/2020	Draft	ToC	Serafeim Moustakidis (AiDEAS)
0.20	22/05/2020	Draft	First draft of the deliverable	Serafeim Moustakidis (AiDEAS)
0.30	09/06/2020	Draft	Final deliverable draft ready to interval review	Serafeim Moustakidis (AiDEAS)
0.40	16/06/2020	Draft	Review comment received	Yannis Nikoloudakis (HMU), Dimitris Apostolakis (FINT)
0.50	23/06/2020	Draft	Final draft sent to the Coordinator for proof reading	Serafeim Moustakidis (AiDEAS)
0.60	25/06/2020	Pre-Final	Quality Control	George Doukas (NTUA), Michael Kontoulis(NTUA)
1.00	25/06/2020	Final	Final	Christos Ntanos (NTUA)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Executive Summary

This deliverable presents the overall development status of the Machine Learning Intrusion Detection (MLID) component on M18 of the project's lifetime and the end of the first interim of MLID's two-staged development phases (M10-M18, M22-M30). This is a versioned document and describes the progress of the development of the first prototype of the component. Within the first development phase of MLID, feature exploration has been performed and a list of the most informative features (reflecting different aspects of users' behaviour) has been identified. Three AI pipelines for intrusion detection have been designed, developed and evaluated in an extensive comparative analysis that includes multiple variants of each pipeline with numerous machine leaning (ML) and deep learning (DL) models.

Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table of Abbreviations

ML	Machine Learning
DL	Deep Learning
MLID	Machine Learning Intrusion Detection
DSS	Decision Support System
DoS	Denial of Service
U2R	User to Root
R2L	Remote to Local
IDS	Intrusion Detection System
AID	Automated Intrusion Detection
BN	Bayesian Networks
CRC	Classification Regression Trees
IG	Information Gain
CNN	Convolutional Neural Networks
GAN	Generative Adversarial Networks
SMOTE	Synthetic Minority Over-sampling TEchnique
SVM	Support Vector Machines
Vec2im	VEctors to IMages
KNN	k-Nearest Neighbor algorithm
DT	Decision Trees
RFE	Recursive Feature Elimination
SCNN	Siamese Convolutional Neural Networks
LDA	Linear Discriminant Analysis
FS	Feature selection

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

1 Introduction

1.1 Purpose & Scope

The scope of this report is to present the current status of the development of the MLID component. The MLID component's development phase is at the end of its first iteration (M10-M18). Three ML pipelines for intrusion detection on honeypots data have been designed, implemented, and tested. This report demonstrates the effectiveness of three proposed pipelines (two supervised and one unsupervised) via a thorough comparative experimentation that involves numerous feature exploration, data mining and machine/deep learning algorithms. The ultimate objectives of this report are to:

i) Identify the pros / cons of the proposed AI pipelines
ii) Identify which feature engineering approach is the most appropriate in our analysis
iii) Quantify the effect of various ML models on each one of the three pipelines. The obtained optimized AI methodologies will be finally used to process the data coming from the honeypot component. In the second development iteration of the MLID component, the obtained AI pipelines will be finetuned to meet the requirements of the incoming honeypot data.

1.2 Structure of the deliverable

The rest of this deliverable is structured as follows. Section 2 presents the main characteristics of the dataset that was employed to guide the design phase of the MLID component. the component's role within the SPHINX ecosystem, its design principles, and its technical characteristics. Section 3 provides a literature review on the use of machine learning for intrusion detection, the main concepts underpinning the proposed methodologies along with a detailed presentation of the three proposed SPHINX ML-empowered intrusion detection pipelines. Section 4 presents and discusses the results of the proposed methodologies and finally, Section 5 concludes this report by presenting a short summary and general conclusion deriving from this report.

1.3 Relation to other WPs & Tasks

This report is closely related to WP4 and more specifically to task 4.4 (The SPHINX AI Honeypots). The Honeypot component of SPHINX will be used to collect interaction data generated by attackers. The generated honeypot data will be sent to MLID, which classifies this data into different categories (classes). The MLID will then send the low-level classification outputs: (i) back to the honeypot, which will take immediate actions (if needed) and (ii) to the Decision Support System (DSS) of SPHINX (as presented in Task 5.1 Decision-Support Application), which will further process the received information and will transform it into actionable rules. Finally, the outcomes of MLID will be communicated to the knowledge base of SPHINX (KB component which is presented in Task 5.5 Common Cyber Security Toolkit).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2 Dataset presentation and analysis

2.1 Dataset description

To validate the performance of the proposed feature extraction methodology, the NSL-KDD dataset[1] was employed. NSL-KDD is actually a variant of the KDD99 dataset, which is the most widespread IDS benchmark dataset at present. Overcoming the limitations of KDD99 (severely unbalanced data, many duplicated and redundant records), NSL-KDD represents a more balanced dataset with a moderate number of records that has allowed the application of various IDS in a large number of papers whose results are consistent as well as comparable. For the aforementioned reasons, we used it as a benchmark in our analysis.

Our dataset consists of 41 features that are categorized in the following four subsets: basic, content, host-based statistical, and time-based statistical features.

Within the data set, 4 different classes of attacks exist: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). A brief description of each attack can be seen below:

- *DoS* is an attack that tries to shut down traffic flow to and from the target system. The IDS is flooded with an abnormal amount of traffic and requests, which the system cannot handle, and shuts down to protect itself. This prevents normal traffic from visiting a network. An example of DoS could be an online retailer getting flooded with online orders on a day with a big sale, and because the network cannot handle all the requests, it will shut down preventing paying customers to purchase anything. This is the most common attack in the data set.
- *Probe or surveillance* is an attack that tries to get information from a network. The goal here is to act like a thief and steal important information, whether it be personal information about clients or banking information.
- *U2R* is an attack that starts off with a normal user account and tries to gain access to the system or network, as a super-user (root). The attacker attempts to exploit the vulnerabilities in a system to gain root privileges/access.
- *R2L* is an attack that tries to gain local access to a remote machine. An attacker does not have local access to the system/network, and tries to "hack" their way into the network.

It is evident from the descriptions above that DoS attacks act differently from the other three attacks; DoS attempts to shut down a system by stopping traffic flow altogether, whereas the other three attacks, attempt to quietly infiltrate the system undetected.

In the table below, a breakdown of the different subclasses of each attack that exists in the data set is shown:

[1] https://www.unb.ca/cic/datasets/nsl.html

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

Table 1 *Breakdown of the different subclasses of each attack that exist in the database*

2.2 Feature characteristics

The data set contains 43 features per record; the first 41 of them are related to the traffic input itself and the last two are the Label (indicating whether it is a normal or attack) and the Score (i.e. the severity of the traffic input itself).

The features can be broken down into four categories: *Intrinsic*, *Content*, *Host-based*, and *Time-based*. Below is a description of the different categories of features:

- *Intrinsic features* can be derived from the header of the packet without looking into the payload itself and hold the basic information about the packet. This category describes features 1–9.
- *Content features* hold information about the original packets, as they are sent in multiple pieces rather than one. With this information, the system can access the payload. This category describes features 10–22.
- *Time-based features* hold the analysis of the traffic input over a two-second window and contains information like how many connections it attempted to make to the same host. These features are mostly counts and rates rather than information about the content of the traffic input. This category describes features 23–31.
- *Host-based features* are similar to the time-based features, except instead of analysing over a 2-second window, it analyses over a series of connections made (how many requests made to the same host over *x*-number of connections). These features are designed to access attacks, which span longer than a two-second window time-span. This category describes features 32–41.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

no	Feature	Description
1	Duration	length (number of seconds) of the connection
2	Protocol type	type of the protocol, e.g. tcp, udp, etc
3	Service	network service on the destination, e.g., http, telnet, etc.
4	Flag	normal or error status of the connection
5	Source bytes	number of data bytes from source to destination
6	Destination bytes	number of data bytes from destination to source
7	Land	1 if connection is from/to the same host/port; 0 otherwise
8	Wrong fragment	number of wrong fragments
9	Urgent	number of urgent packets
10	Hot	number of hot indicators
11	Number failed logins	number of failed login attempts
12	Logged in	1 if successfully logged in; 0 otherwise
13	Num compromised	number of compromised conditions
14	Root shell	1 if root shell is obtained; 0 otherwise
15	Su attempted	1 if su root command attempted; 0 otherwise
16	Num root	number of root accesses
17	Num file creations	number of file creation operations
18	Num shells	number of shell prompts
19	Num access files	number of operations on access control files
20	Num outbound cmds	number of outbound commands in an ftp session
21	Is host login	1 if the login belongs to the hot list; 0 otherwise
22	Is guest login	1 if the login is a guest login; 0 otherwise
23	Count	number of connections to the same host as the current connection in the past two seconds
24	Srv count	number of connections to the same service as the current connection in the past two seconds
25	Serror rate	number of connections to the same host as the current connection in the past two seconds
26	Srv error rate	% of connections that have SYN errors
27	Rerror rate	% of connections that have REJ errors
28	Srv error rate	% of connections that have REJ errors
29	Same srv rate	% of connections to the same service
30	Diff srv rate	% of connections to different services
31	Srv diff host rate	% of connections to different hosts
32	Dst host count	Number of connections to the same host to the destination host as the current connection in the past 2 seconds
33	Dst host srv count	Number of connections from the same service to the destination host as the current connection in the past 2 seconds
34	Dst host same srv rate	% of connections from the same services to the destination host
35	Dst host diff srv rate	% of connections from the different services to the destination host
36	Dst host same src port rate	% of connections from the port services to the destination host
37	Dst host srv diff host rate	% of connections from the different hosts from the same service to the destination host
38	Dst host error rate	% of connections that have SYN errors from the same host to the destination host
39	Dst host srv error rate	% of connections that have SYN errors from the same service to the destination host
40	Dst host error rate	% of connections that have REJ errors from the same host to the destination host
41	Dst host srv error rate	% of connections that have REJ errors from the same service to the destination host

Table 2 *List of features referring to traffic input*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Specifically, a 20% subset of the NSL-KDD training data (the NSL-KDD Train 20 variant), that is already available in the literature, was used in this deliverable comprising of 25,192 data points, whereas the NSL-KDD Test + data file (comprising of 22,544 data points) was utilized for testing. Given that the focus of the MLID is on the discrimination between attacks and normal data, the intrusion detection problem was considered as binary by merging all the anomalous records (categories 1-4) into one class.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3 SPHINX ML-based intrusion detection framework

3.1 Literature Review

An intrusion detections system (IDS) is a security tool that collects information from various sources (e.g. routers, computers, network data) aiming at identifying malicious activities and/or users that attempt to either get access to computers, steal protected data or even manipulate and disable information systems [1]. IDSs can be categorized into three main categories [2]. The first category of IDSs compares the collected patterns of network traffic with specific and pre-determined signatures (attack patterns). An attack is detected once there is match with an already known pattern, however this kind of IDS is incapable of identifying new (unknown) malicious activities. The second category builds on a set of rules and thresholds (specifications) that have been manually specified by security experts. These specification-based IDSs do not generate false alarms when unusual (but legitimate) program behaviours are encountered but in general the specifications development is a tedious and expensive process while the specified set of rules is often very difficult to evaluate and verify. Unlike signature and specification IDSs, Automated Intrusion Detection (AID) systems are a new category that employs machine learning, statistical-based or knowledge-based methods to define a normal model of the behaviour of a computer system. The effectiveness of AID systems depends a lot on the quantity as well as quality of the network traffic patterns that are used as data instances during their training.

In the last few decades, ML has been used to improve intrusion detection [3]. There is a large number of related studies using various synthetic datasets (such as KDD-Cup 99 or DARPA 1999 datasets) to develop and validate ML-empowered AID systems. Any significant deviation between the observed 'normal' behaviour can be regarded as an anomaly, which can be then interpreted as an intrusion. The main assumption of the aforementioned approaches is that malicious behaviour differs from typical user behaviour. One simplistic method to decide whether a behaviour is normal or abnormal is by comparing it with the standard deviation of the normal users' behaviour in the training dataset. Any example exceeding the pre-determined threshold (e.g. three times the standard deviation) could be classified in the intrusion category. ML provides a more sophisticated method for decision making overcoming the deficiencies of the heuristic approaches (such as the manual selection of the threshold etc.). Development of ML-based AID systems comprises two phases: the training phase and the testing phase.

a. In the training phase, the normal traffic profile is used to learn a model of normal behaviour,
b. In the testing phase, a new data set is used to validate the system's.

AIDs can be classified into several categories based on the method used for training, for instance, statistical based, knowledge-based and machine-learning-based [4]. The main advantages of ML-empowered AID systems are: (i) Their ability to identify zero-day attacks without relying on a signature database [5]. A danger signal can be triggered when the examined behaviour differs from the usual behaviour. (ii) Their capability to discover internal malicious activities. An alarm will be created in cases where an intruder starts making transactions in a compromised account that deviate from the typical user activity. (iii) The normal user behaviour is hidden to intruders and thus it becomes more difficult for them to remain undetected.

The objective of using machine learning techniques is to create IDSs with improved accuracy and less requirement for prior human knowledge. However, one of the main challenges of current AIDs is the high false positive rates because anomalies may just be new normal activities rather than genuine intrusions.

One of the crucial phases in today's ML pipelines is the process of extracting knowledge from large quantities of data. To effectively extract knowledge from raw data, ML relies on a set of rules, methods, or complex "transfer functions" that are applied to find interesting data patterns, or to recognize and predict behaviour [6]. Many ML algorithms (such as clustering, neural networks, association rules, decision trees, genetic

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

algorithms, and nearest neighbour methods) have been recently applied in the area of AIDs for discovering knowledge from intrusion datasets ([7],[8]). Some prior research in data mining has examined the use of different algorithms to extract meaningful information for intrusion data. Two feature selection algorithms were investigated by Chebrolu et al. 2015 employing Bayesian networks (BN) and Classification Regression Trees (CRT) [9]. The outputs of the aforementioned algorithms were finally combined to increase accuracy. Bajaj et al. 2013, proposed a technique for feature selection using a hybrid approach that combines Information Gain (IG) and correlation attribute evaluation [10]. To validate the discrimination capacity of the selected features, the authors applied several classification algorithms such as C4.5, naïve Bayes, NB-Tree and Multi-Layer Perceptron [11]. Genetic-fuzzy rule mining has been also explored to evaluate the importance of IDS features by Elhag et al. 2015 [12]. Thaseen and Kumar 2013 proposed a Random Tree model to improve the accuracy and reduce the false alarm rate [13], whereas Subramanian et al. also studied the performance of decision tree algorithms on the NSL-KDD dataset [14].

Unlike ML approaches that require the extraction of features, Deep learning (DL)-based detection methods learn features automatically, in an end-to-end fashion (directly from raw data to decisions). DL is gradually attracting more interest in AID studies. A Convolutional Neural Network (CNN) CNN-based AID methodology was presented by Potluri et al. 2018, conducting experiments on the NSL-KDD and the UNSW-NB datasets [15]. In the pre-processing phase, the features of the datasets were transformed into images of 8*8 pixels. Then, a three-layer CNN was trained to classify the attacks. Pre-trained deep networks (ResNet 50 and GoogLeNet) were also explored as alternative solutions to the task of extracting new informative features. The proposed CNN performed best, reaching accuracies of 91.14% on the NSL-KDD and 94.9% on the UNSW-NB 15. A sparse autoencoder was also proposed by Zhang et al. 2018 to extract features from the NSL-KDD dataset [16]. The extracted features were supplied to an XGBoost model with the objective to detect attacks. To overcome the observed data imbalance problem, data resampling was employed (using SMOTE). The SMOTE algorithm oversamples the minority classes and divides the majority classes into many subclasses so that every class is balanced. Data augmentation with GANs has been also explored by Zhang et al. 2019 [17]. The GAN model was used to generate data similar to the flow data of KDD99. Adding this generated data to the training set increased the generalization capacity of the detection model that was able to identify not only attacks but attack variants as well.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3.2 Concepts underpinning the proposed methodologies

3.2.1 Feature exploration

Feature exploration is the initial step in data analysis, where users explore a large data set in an unstructured manner to uncover initial patterns, characteristics, and points of interest. This process is not meant to reveal every bit of information a dataset holds, but rather to help create a broad picture of important trends and major points to study in greater detail. This process makes deeper analysis easier because it can help in the targeting of future searches, and initiates the process of excluding irrelevant data points and search paths that may return no results. In this section, a hybrid feature exploration approach combining multiple FS algorithms to avoid bias was implemented.

Hybrid combining multiple FS algorithms to avoid bias

A hybrid feature selection methodology was employed consisting of filter, wrapper, and embedded techniques, whereas feature ranking was decided based on a majority voting system. Applying each technique separately, the order of the feature importance emerged from the frequency of feature appearance in the selection criteria. The features were ranked with respect to the votes received.

A short overview of the feature selection algorithms investigated here, is given below.

Filter algorithms:

Pearson Correlation is the most important correlation factor and relates to quantitative variables, while based on the concept of linear relationship. If there is a linear dependence between the two features, then their correlation coefficient is ± 1. If there is no dependence, the correlation coefficient is 0. In this case, there is no linear relationship between the two factors. However, if two variables are highly correlated among themselves, they provide redundant information regarding the target. Consequently, the second variable does not add additional information, so removing it can help to reduce the dimensionality. In this approach, we set the maximum number of the selected features to be 30 [18].

Chi-squared independence test [19] was applied to examine the relationship between two quality variables. The Chi-squared statistical test also works manually with non-negative numerical and quantitative characteristics. The specific test compares the degree of agreement (or correlation) between the theoretical frequency and the actual frequency. The algorithm was decided to terminate at 30 selected features. The termination criterion was manually selected after a trial-and-error exploration process.

Wrapper algorithms:

Recursive Feature Elimination (RFE) [20] is a greedy optimization algorithm which aims to find the best performing feature subset. In each iteration, it creates models and keeps aside the best or the worst performing feature. Each next model includes reduced number of features until all the features are exhausted. At the end, it ranks the features based on the order of their elimination. In this approach, the logistic regression classifier was selected to drive the elimination process whereas the termination criterion was also set to 30 features.

Embedded FS techniques:

Logistic regression (L2 penalty) is an embedded method relying on regularized logistic regression models. Furthermore, this approach is based on small subsets of the full feature space by sampling this space at random. The sampling probability depends on the estimated feature relevance. In addition, the initial relevance of each feature is estimated according to a t-test ranking [21].

Random forests are a popular method for feature ranking, due to the fact that they require very little feature engineering and parameter tuning. But they come with their own limitations, especially when data

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

interpretation is concerned. In highly correlated data, strong features can end up with low scores and the method can be biased towards variables with many categories. In addition, this method stochastically rearranges all values of the features for each tree and uses the RF model to predict this permuted feature [22].

Light GBM is a gradient boosting framework that uses tree-based learning algorithm. Specifically, the Gradient-based One-Side Sampling and Exclusive Feature Bundling are used to deal with large number of data instances and large number of features. Light GBM can handle the large size of data and takes lower memory to run and is faster than Gradient Boosting Decision Tree [23].

The proposed feature selection proceeds along the following steps:

```
Step 1: All features were normalized as described in Section 3.2
Step 2: We performed each one of the six FS techniques separately resulting to the creation of the following six feature subsets FSSi, i=1…,6
Step 3: Main loop
        Step 3.1 For each feature j, we set Vj=0, j=1,...,M where M the total number of features
        Step 3.2 Set j=1
        Step 3.3 if a feature j is selected in FFSi, then Vj=Vj+1;
        Step 3.4: We repeat step 3.2 for each one of the six FS techniques
        Step 3.5 Set j=j+1 and return to step 3.2
        Terminate main loop when j>m
Step 4: Rank features to descending order with respect to Vj
end
```

3.2.2 Machine learning models

Various ML models were evaluated for their suitability in solving the intrusion detection problem. A brief description of these models is provided below.

XGboost is a popular and efficient implementation of the Gradient Boosted Trees algorithm. It is a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques. Specifically, this model is a sum of CART (tree) learners that try to minimize the log loss objective and the scores at leaves. These scores are actually the weights that have a meaning as a sum, across all the trees of the model. Furthermore, they are always adjusted in order to minimize the loss [24].

Random Forest classifier is an ensemble algorithm. Ensemble algorithms are those that combine more than one algorithm of same or different kind for classifying objects. The random forest classifier creates a set of decision trees from randomly selected subsets of a training set. It then aggregates the votes from different decision trees to decide the final class of the test object [25].

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. DTs are simple to understand and to interpret. They require little data preparation and perform well even if their assumptions are somewhat violated by the true model from which the data were generated [26].

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Naive Bayes learners and classifiers can be extremely fast. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution [27].

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier's detection. They are effective in high dimensional spaces and still effective in cases where the number of dimensions is greater than the number of samples. Furthermore, SVMs use a subset of training points in the decision function, called support vectors [28].

K-Nearest Neighbor (KNN) is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. Similarity is defined according to a distance metric between two data points. A popular one is the Euclidean distance method [29].

Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems. The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression, since the outcome in logistic regression is a probability between 0 and 1 [30].

AdaBoost is a generic iterative supervised learning algorithm that combines weak hypotheses into a much more accurate master hypothesis. This master hypothesis H is a weighted linear combination of these hypotheses. The master hypothesis typically performs much better than any of the weak hypotheses alone, and thus is likely to predict better on new examples as well. Although the original AdaBoost algorithm creates binary classifiers, there are a number of variants designed for multiple classes [31].

Linear discriminant analysis (LDA) refers to the classifier design. Given a number of variables as the data representation, each class is modelled as Gaussian (with a covariance matrix and a mean vector). Observations are now classified to the class of the nearest mean vector according to Mahalanobis distance. The decision surfaces between classes become linear if the classes have a shared covariance matrix. In this case the decision surfaces are called Fisher discriminants, and the procedure of constructing them is called Linear Discriminant Analysis [32].

Hyperparameter selection was implemented to optimize the performance of our models and to avoid the overfitting and the bias error. Each model was optimized with respect to a number of parameters. Specifically (i) gamma, maximal depth, minimum child and weight were optimized for XGboost, (ii) criterion, minimum samples leaf, minimum samples split and number of estimators for Random Forest, (iii) maximal features, minimum samples and minimum number of decision splits for Decision Trees, (iv) C and kernel for SVMs, (v) leaf size and k-parameter for KNN and (vi) penalty and C for Logistic Regression.

3.2.3 Deep learning

Deep learning architectures are employed in pipelines B and C. A short theoretical background of the selected deep learning approaches is given in the subsections below.

A. Siamese convolutional neural networks

A Siamese network [33] is a particular neural network architecture consisting of two identical sub-convolutional networks, which is used in a weakly supervised metric learning setting. The goal of the network is to make the output vectors similar if input pairs are labelled as similar, and dissimilar for the input pairs that are labelled as dissimilar. Recently, the Siamese network has been applied to speech feature classification [34], text classification [35], wireless positioning and channel charting [36], multi-class classification of Alzheimer disease [37] and remote sensing scene classification [38].

Recently deep learning has achieved great success on many computer-vision tasks. Specifically, CNN has set records on standard object recognition benchmarks [39]. With a deep structure, the CNN can effectively learn complicated mappings from raw images to the target, which requires less domain knowledge compared to handcrafted features and shallow learning frameworks.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

CNN is a multilayer learning framework, which consists of an input layer, a few convolutional layers, and fully connected layers, as well as an output layer on which the loss function is defined. The goal of CNN is to learn a hierarchy of feature representations. Signals in each layer are convolved with several filters and further down sampled by pooling operations, which aggregate values in a small region by functions including max, min, and average. The learning of CNN is based on Stochastic Gradient Descent [40].

Siamese Convolutional Neural Network has been used successfully for dimension reduction in weakly supervised metric learning. Instead of taking single sample as input, the network typically takes a pair of samples, and the loss functions are usually defined over pairs. A typical loss function of a pair has the following form:

$$L(s_1, s_2, y) = (1 - y)aD_w^2 + y\beta \mathrm{e}^{\gamma \mathrm{D_w}} \tag{1}$$

where:

- s_1 and s_2 are two samples,
- y is the binary similarity label,
- $D_w = \|f(s_1; w_1) - f(s_2; w_2)\|_1$ is the distance, which is based on a defined distance function (normalized L1) (Chopra et al., 2005).

This can be regarded as a metric learning approach. Unlike methods that assign binary similarity labels to pairs, the network aims at bring the output feature vectors closer for input pairs that are labelled as similar, or push the feature vectors away if the input pairs are labelled as dissimilar.

This can be regarded as a metric learning approach. Unlike methods that assign binary similarity labels to pairs, the network aims at bring the output feature vectors closer for input pairs that are labelled as similar, or push the feature vectors away if the input pairs are labelled as dissimilar. The Siamese network is frequently illustrated as two identical networks for two different samples. In each Stochastic Gradient Descent iteration, pairs of samples are processed using two identical networks, and the error computed by equation (1) is then back-propagated and the gradients are computed individually base on the two sample sets. The Siamese network is updated by the average of these two gradients [41].

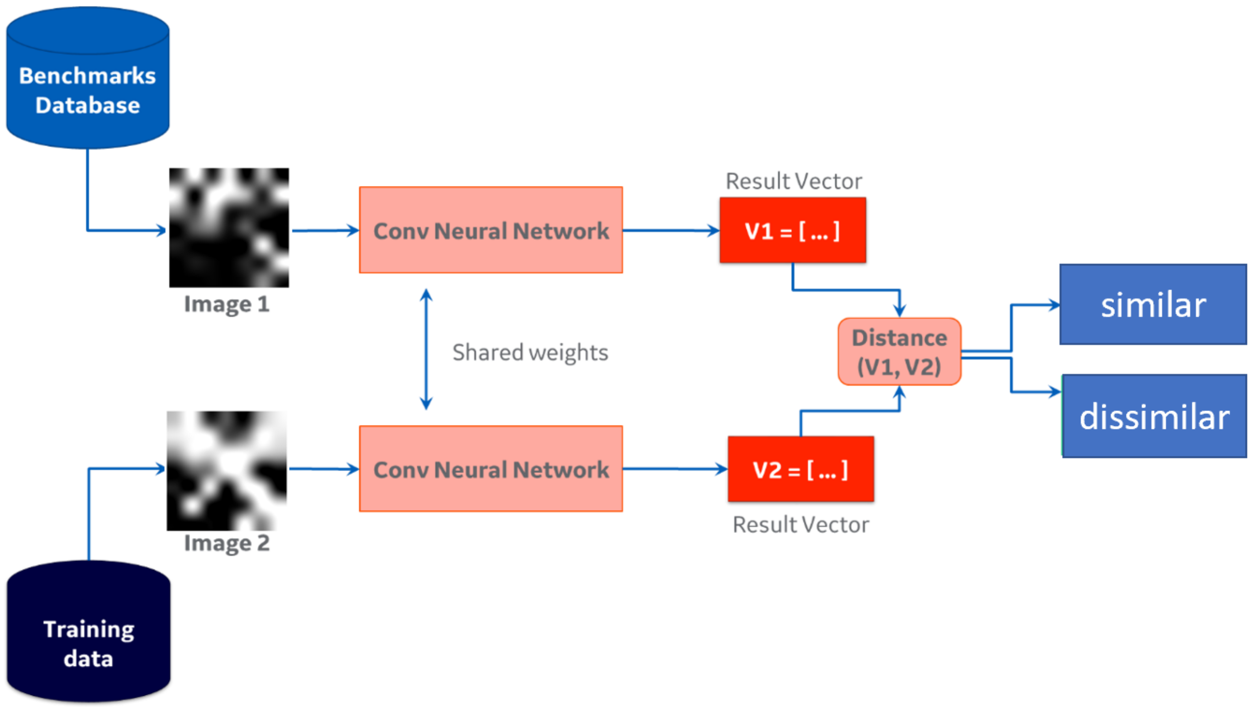


Figure 1 *Dimension reduction using Siamese network.*

B. Autoencoders

A deep autoencoder is a feed-forward multi-layer neural network in which the desired output is the input itself [42]. Upon first glance, this process may seem trivial since the identity mapping would have no reconstruction error. However, autoencoders become non-trivial when the identity map is disallowed either by way of some type of regularization or, more importantly for the current derivation, by having hidden layers which are a low-dimensional, non-linear representation of the input data. In particular, autoencoders learn a map from the input to itself through a pair of encoding and decoding phases:

$$\bar{X} = D\big(E(X)\big) \tag{2}$$

where X is the input data, E is an encoding map from the input data to the hidden layer, D is a decoding map from the hidden layer to the output layer, and $\bar{X}$ is the recovered version of the input data. The idea is to train E and D to minimize the difference between X and $\bar{X}$.

In particular, an autoencoder can be viewed as a solution to the following optimization problems:

$$\min_{D,E} \left\| X - D\big(E(X)\big) \right\| \tag{3}$$

where ||·|| is commonly chosen to be the l2-norm.

Usually, an autoencoder with more than one hidden layers is called a deep autoencoder and each additional hidden layer requires an additional pair of encoders E(·) and decoders D(·). By allowing many layers of encoders

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

and decoders, a deep autoencoder can effectively represent complicated distributions over the input X. In the sequel, our focus will be on deep autoencoders with all autoencoders assumed to be deep [43].

To avoid trivial lookup table-like representations of hidden units, autoencoders reduces the number of hidden units. Autoencoders with various other regularization has also been developed. Contractive autoencoders use gradients of activations as penalty term and try to model data with sparse activations that only respond to the true nature of the data [44]. Denoising autoencoders uses a adds noise to the original input vector x and uses this noisy input $\hat{x}$ as the input vector. The difference between the resulting output, the reconstruction of the noisy input, and the original input is used as the reconstruction error. In short, this is training the autoencoder to reproduce the original input x from a noisy input $\hat{x}$. This allows the autoencoder to be robust to data with white noise and capture only meaningful patterns of the data [45].

Autoencoder based anomaly detection is a deviation-based anomaly detection method using semi-supervised learning. It uses the reconstruction error as the anomaly score. Data points with high reconstruction are considered to be anomalies. Only data with normal instances are used to train the autoencoder. After training, the autoencoder will reconstruct normal data very well, while failing to do so with anomaly data which the autoencoder has not encountered. The anomaly detection algorithm is using reconstruction errors of autoencoders [46].

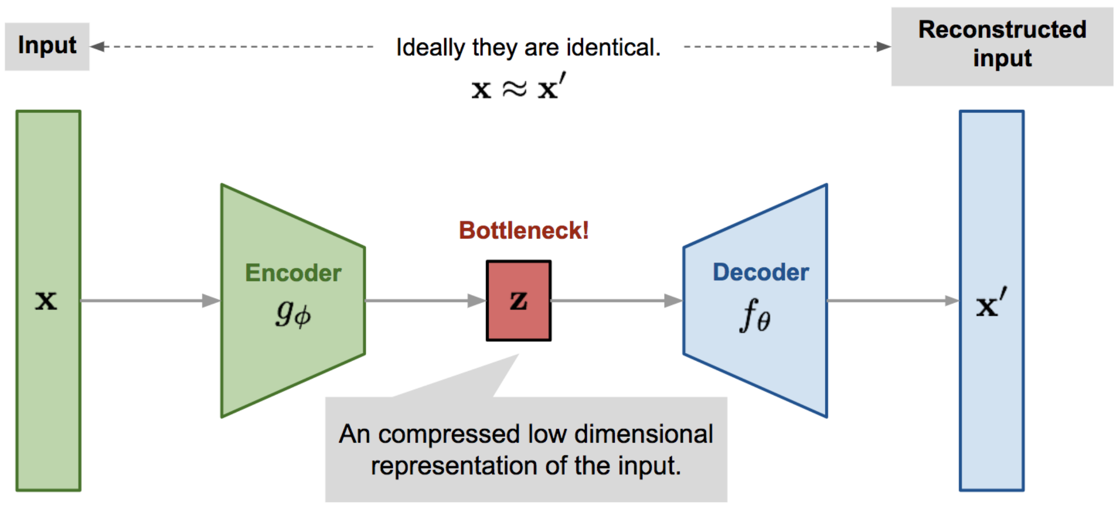

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

***Figure 2** Autoencoder concept graph*

3.2.4 Key Performance Indicators

The following criteria were used in our analysis to evaluate the effectiveness of the proposed pipelines:

A. Overall accuracy

Overall accuracy is the probability that an individual will be correctly classified by a test; that is, the sum of the true positives plus true negatives divided by the total number of individuals tested. The overall accuracy is calculated by summing the number of correctly classified values and dividing by the total number of values. The correctly classified values are located along the upper-left to lower-right diagonal of the confusion matrix. The total number of values is the number of values in either the truth or predicted-value arrays.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

B. Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

where:

- TP: True Positives
- FP: False Positives
- FN: False Negatives
- TN: True Negatives

C. Per class accuracies

Per class accuracies can be extracted from the confusion matrix, but they can be misleading if they have not been differentiated into producers' and users' accuracy. It also referred to as errors of omission and errors of commission respectively. Errors of omission calculate the probability that a reference sample has been classified correctly. Furthermore, errors of commission calculate the probability that a sample from the classified data actually represents that category on the ground.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

3.3 Proposed methodology and variants

3.3.1 Pipeline A: Machine learning pipeline for intrusion detection

A machine learning pipeline was initially designed and developed to identify intrusion patterns in the selected data classification problem. The proposed methodology is depicted in Figure 3 and comprises five (5) processing phases that are presented below.

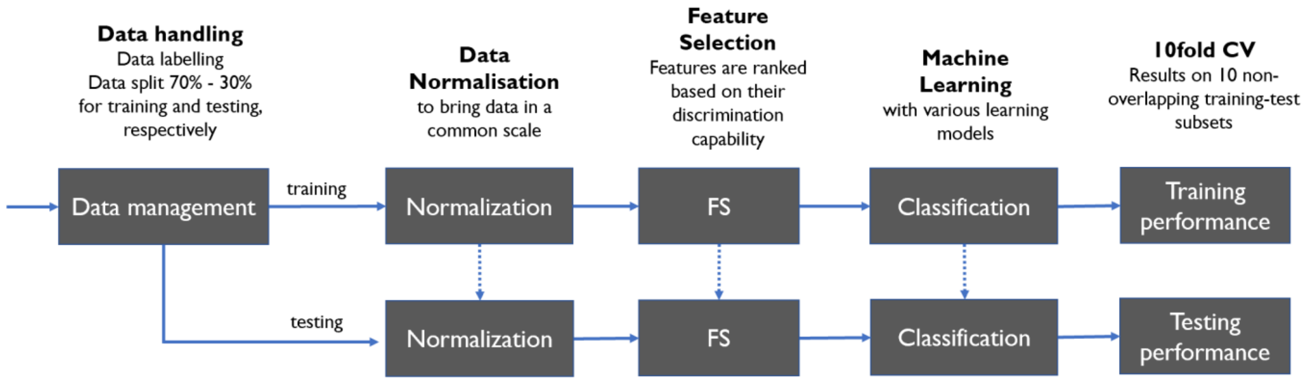

Data handling
Data labelling
Data split 70% - 30% for training and testing, respectively
Data Normalisation
to bring data in a common scale
Feature Selection
Features are ranked based on their discrimination capability
Machine Learning
with various learning models
10fold CV
Results on 10 non-overlapping training-test subsets
Data management
training
testing
Normalization
FS
Classification
Training performance
Normalization
FS
Classification
Testing performance

Figure 3 *Proposed pipeline A*

A. Data preparation and handling

The NSL-KDDTrain20 data file (comprising of 25,192 data points) was used for training and the NSL-KDD Test+ data file (comprising of 22,544 data points) was utilized for testing. Moreover, simple data encoding techniques were employed to convert unstructured, textual data into numeric representations which can then be understood by machine learning algorithms (categorical values).

B. Normalization

Z-score was used to normalize features to a common scale (having a mean of zeros and a standard deviation of 1). Specifically, the data points $x_{i,j}$ were standardized with the following formula:

$$z_{i,j} = \frac{x_{i,j} - \overline{x_j}}{s_j} \tag{4}$$

where:

- $x_{i,j}$ denotes the feature *j* of data sample x_i
- $\bar{x}_j$ denotes the mean value of feature *j*
- s_j is the standard deviation of feature *j*
- $z_{i,j}$ denotes the normalized version of $x_{i,j}$

C. Feature Selection

A feature selection approach was employed to identify the most informative features and rank them in order of significance (specifically, the hybrid FS technique that combines the outcomes of multiple well-known FS models to avoid bias) The specifics of this approach are given in Section 3.2.1 B.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

D. Machine learning

Six (6) machine learning techniques were investigated for their suitability in identifying intrusion patterns using the selected features (as generated in the previous phase C):

- AdaBoost
- Random Forest
- Support Vector Machines
- Nearest neighbor classifier
- Decision Trees
- Discriminant analysis

Hyperparameter selection was performed to optimize the performance of the ML models. A validation subset was held out from the training set (a randomly selected 10%) as a criterion for selecting the optimum hyperparameters by means of a grid search process.

E. Validation

The discrimination capacity of the proposed pipeline was performed on the testing dataset.

3.3.2 Pipeline B: Novel feature dimensionality reduction empowered by deep learning towards the extraction of informative risk indicators

The proposed pipeline B includes four processing steps: (i) data pre-processing making use of a fuzzy allocation scheme to convert raw data into fuzzy values, (ii) a data transformation technique that generates images comprising of fuzzy memberships, (iii) a novel feature extraction algorithm employing Siamese convolutional neural networks and finally (iv) a learning process for training, and evaluation of the results, as illustrated in Figure 4. The proposed methodology is thoroughly presented in the following sections.

This deliverable contributes to current AID systems by transforming the available multi-dimensional network traffic data into an actionable and easy-to-understand indicator for security experts. This has been achieved by designing a novel feature extraction pipeline that builds on the latest advances of data mining and ML/DL. Specifically, a fuzzy allocation scheme is initially applied to transform raw data to fuzzy class memberships. Then a data transformation mechanism converts feature vectors to images (Vec2im) and a dimensionality reduction module makes use of Siamese convolutional neural networks to reduce the input data dimensionality into a 1-d feature space. The performance of the proposed pipeline was validated via a thorough comparative analysis that demonstrated its effectiveness over a number of well-known feature selection and extraction techniques.

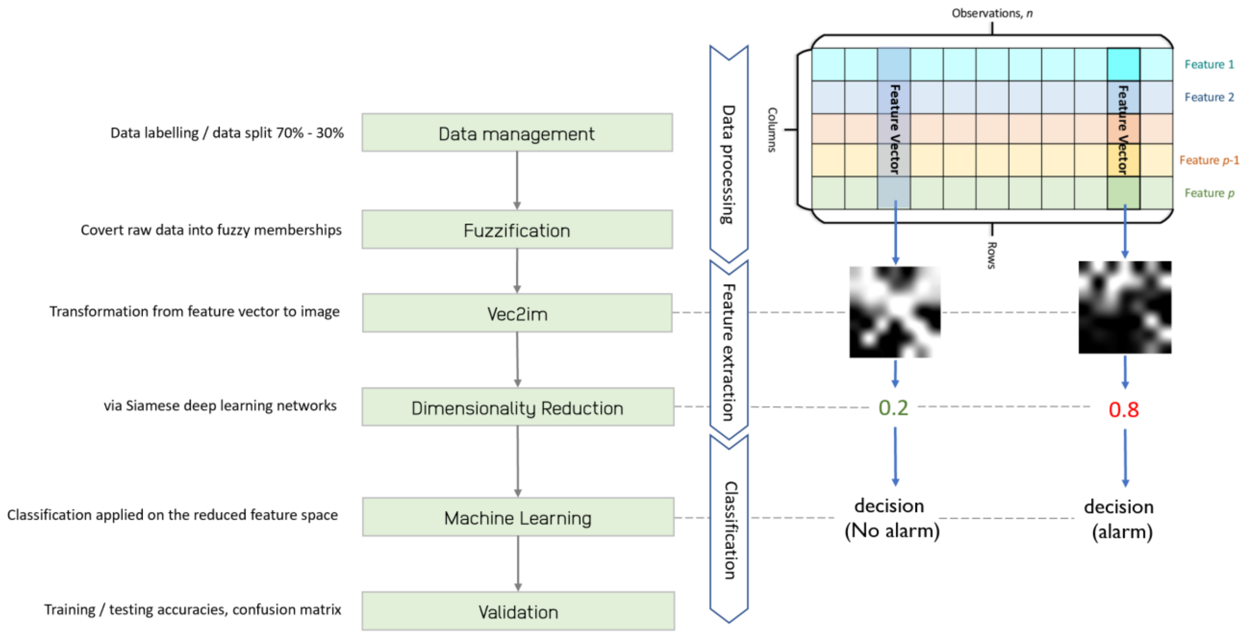


***Figure 4** Proposed pipeline B*

A. Data preparation

Specifically, a 20% subset of the NSL-KDD training data (the NSL-KDD Train 20 variant) was used in our paper comprising of 25,192 data points, where the NSL-KDD Test+ data file (comprising of 22,544 data points) was utilized for testing. Moreover, simple data encoding techniques were employed to convert unstructured, textual data into numeric representations which can then be understood by machine learning algorithms (categorical values).

B. Fuzzification and image formulation (Vec2im)

First of all, the non-numeric attributes of the dataset were converted into numeric values. For the efficient training of machine learning algorithms, input data is typically transformed by a number of pre-processing routines with data normalization being the gold standard. Different algorithms could be used to normalize the input data (such as min-max normalization or normalization with respect to standard deviation), however in this paper we employed a fuzzy allocation scheme as described below.

Fuzzification: To normalize as well as evaluate the classification capabilities of each feature, we applied a simple fuzzy allocation scheme that assigs varying degrees of patterns to every class. For feature *j*, the fuzzy membership $u_i(x_{k,j}) \in [0,1]$ indicating the degree to which $x_{k,j}$ belongs to class *i* is determined by:

$$u_i(x_{k,j}) = \frac{1}{\sum_{m=1}^{c}\left[\frac{(x_{k,j}-u_{i,j})^2}{(x_{k,j}-u_{m,j})^2}\right]^{1/(b-1)}} \quad (5)$$

where $u_{i,j} = \sum_{k \in A_i} x_{k,j}/N_i$ is the class *i* mean along the $x_{k,j}$ component, A_i is the set of indexes of the training examples belonging to class *i* , N_i is the number of class *i* patterns and *b* is a fuzzification factor (b = 2 in our experiments). In our paper, every feature component of $x_{k,j}$, was converted to $u_1(x_{k,j})$ that denotes its fuzzy membership to class 1 (normal / non intrusion class). High values of $u_1(x_{k,j})$ close to 1 indicate a strong membership to the non-intrusion class whereas low $u_1(x_{k,j})$ values close to 0 are representative of examples belonging to the malicious class.

Vec2im: At the second phase of processing, the generated memberships were re-placed in a matrix format resulting to one grey-scale image per example. Specifically, the 41 features $x_{k,j}, j = 1, \dots, 41$ were transformed to 41 fuzzy memberships $u_i(x_{k,j}), j = 1, \dots, 41$ and finally a 7×7 image was created per sample by placing the fuzzy memberships in a matrix as presented in Figure 5. Zero values were also included in the matrix in random cells to fill the eight gaps (given that the dimensionality of the initial feature set was 41 with a total of 49 cells to be filled in the matrix). Fuzzy memberships and zero values were ordered randomly since it was concluded that their order has not any significant impact on the final performance of the proposed methodology.

$$\underset{\textbf{Feature vector } 1\times 41}{\left[x_{k,1}, x_{k,2}, \dots, x_{k,41} \right]} \longrightarrow \overset{\textbf{Feature matrix } 7\times 7}{\begin{bmatrix} u_1(x_{k,1}), & u_1(x_{k,2}) & , \dots, & u_1(x_{k,7}) \\ u_1(x_{k,8}), & u_1(x_{k,9}) & , \dots, & u_1(x_{k,14}) \\ \dots & 0 & \dots & \dots \\ 0 & , u_1(x_{k,36}) & , \dots, & u_1(x_{k,41}) \end{bmatrix}}$$

Figure 5 *Converting feature vectors to images (Vec2im)*

C. Dimensionality Reduction with Siamese deep learning networks

Deep Siamese Convolutional Neural Networks (SCNN) architecture is a variant of neural networks that was originally designed to solve signature verification problem of image matching [47]. It has also been used for one-shot image classification [48], face verification where the categories are not known in advance [33] as well as for dimensionality reduction [49]. SCNN consist of two identical symmetric CNN subnetworks that share the same weights. In our experiment, each identical CNN was built using one convolutional layer followed by three fully connected layers. The rectified linear units (ReLU) nonlinearity was applied as the activation function for all layers, and adaptive moment estimation (ADAM) optimizer was utilized to control learning rate [50]. The similarity between images was calculated by Euclidean distance, and the contrastive loss [33] was calculated to define the loss function as follows:

$$\mathcal{L}(W, I_1, I_2) = \mathbf{1}(L=0)\frac{1}{2}D^2 + \mathbf{1}(L=1)\frac{1}{2}[\max(0, margin - D)]^2 \tag{6}$$

$$\text{where } D = \|f(I_1) - f(I_2)\|^2, \tag{7}$$

$I_1$and I_2 are a pair of the generated images fed into each of two identical CNNs. 1(·) is an indicator function to show that whether two images have the same label, where L = 0 represents the images have the same label and L = 1 represents the opposite. W is the shared parameter vector comprising of the weights that both neural networks share each other. $f(I_1)$ and $f(I_2)$ are the latent representation vectors of input I_1 and I_2, respectively and D is the Euclidean distance between them. The selected SCNN architecture (as depicted in Figure 6) reduces the dimensionality of the 41-dimensional feature space to a single 1-d space.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

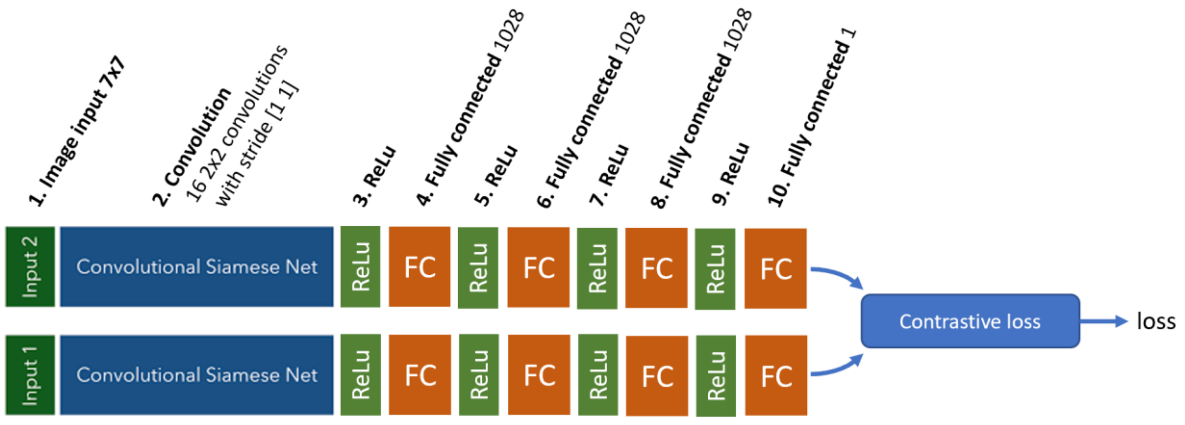


Figure 6 *Selected Siamese network architecture for dimensionality reduction*

D. Decision making on the reduced feature space

To evaluate the discrimination capacity of the extracted features, we employed various machine learning models trained to implement the binary classification task on the resulted 1-d space. We tested linear discriminant analysis (LDA) and Naïve Bayes [51] to provide a baseline for comparisons with more advanced models. We also evaluated decision trees ([52],[53]), driven by Gini's diversity index, KNN, as well as non-linear support vector machines (SVM) algorithms ([54],[55]) with Gaussian kernel, which can deal with the overfitting problems that appear in high-dimensional spaces. The ensemble techniques AdaBoost [56] and Random Forest [57] were also evaluated using DT models as weak learners.

E. Validation

To achieve a fair comparison between the different approaches, hyperparameter selection was performed for each one of the investigated machine algorithms. A validation subset was held out from the training set (a randomly selected 10%) as a criterion for: (i) selecting the optimum hyperparameters by means of a grid search process as well as (ii) deciding the termination of the SCNN learning.

3.3.3 Pipeline C: Unsupervised approach for anomaly detection

In contrast to the well-known classification setup, where training data is used to train a classifier and test data measures performance afterwards, there are multiple setups possible when talking about anomaly detection. Basically, the anomaly detection setup to be used depends on the labels available in the dataset. Unsupervised anomaly detection is the most flexible setup which does not require any labels. The idea is that an unsupervised anomaly detection algorithm scores the data solely based on intrinsic properties of the dataset. Typically, distances or densities are used to give an estimation what is normal and what is an outlier.

To enable label-free intrusion detection, an unsupervised pipeline was also designed relying on the latest advances of deep learning and especially autoencoders. The proposed pipeline (figure 7) is organized in the following steps.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

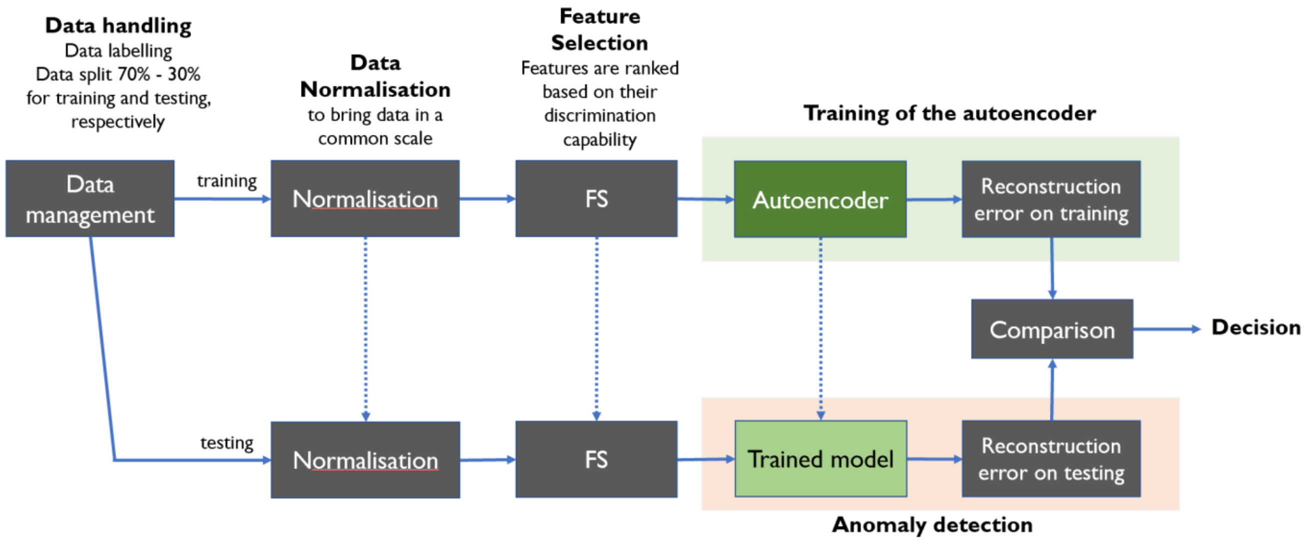

Data handling
Data labelling
Data split 70% - 30% for training and testing, respectively
Data Normalisation
to bring data in a common scale
Feature Selection
Features are ranked based on their discrimination capability
Training of the autoencoder
Data management
training
testing
Normalisation
FS
Autoencoder
Reconstruction error on training
Comparison
Decision
Normalisation
FS
Trained model
Reconstruction error on testing
Anomaly detection

Figure 7 *Proposed pipeline C*

A. Data handling

The NSL-KDD Train 20 data file (comprising of 25,192 data points) was used for training and the NSL-KDD Test+ data file (comprising of 22,544 data points) was utilized for testing. Moreover, simple data encoding techniques were employed to convert unstructured, textual data into numeric representations which can then be understood by machine learning algorithms (categorical values).

B. Data normalization

Z-score was also used here to normalize features to a common scale (having a mean of zeros and a standard deviation of 1).

C. Feature Selection

The hybrid FS technique (as described in Section 3.2.1 B) was applied to provide a ranking of the available features with respect to their expected discrimination capacity. The output of this FS processing step is the creation of training and testing feature subsets of increasing dimensionality. The autoencoder (step D) was applied on the each one of the generated feature subsets and the optimal feature subset was identified.

D. Learning process

An autoencoder was applied on the selected features using only data points of the training set that belong to class 1 (normal activity). The main idea is to use autoencoders to learn the normal behavior of users and then to use them to detect abnormal states (intrusions). After this "normal" data set (class 1) has been obtained the training of the ML model proceeds in unsupervised fashion, without the need of labels. A critical advantage of this method is that it will be able to identify faulty conditions even though these have not been encountered earlier during the training phase. With this method we do not need to inject anomalies (class 2) during the training phase and we do not require intrusion logs or changes to the standard users' behavior. The learning of this methodology can be organized as given below:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

(i) The autoencoder is trained using only data from class 1 (and specifically using the features selected in step C)

(ii) Once constructed the trained model is used to reconstruct data from both Class 1 and Class 2.

(iii) The differences D between the autoencoder's input and output are calculated for each data point.

(iv) The generated D values (that actually declare reconstruction errors) for class 1 training data are grouped together forming the group D_1. The same process is repeated for Class 2 training data points generating the D_2 group.

(v) Higher the reconstruction error, higher the possibility of that data point being an anomaly. Based on this, a classifier is finally applied to separate groups D_1 and D_2 and identify whether an input is an anomaly (possibly an intrusion) or not.

The hyperparameters of the autoencoder (number of layers and nodes per layer) were selected by trial and error.

E. Validation

The testing dataset was supplied to the trained autoencoder model. The obtained reconstruction errors were used as input to the trained classifier that classifies them as anomalies or not. The classification accuracy on the testing set was considered as the final evaluation criterion of the proposed methodology.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4 Results

Section 4 presents the evaluation results of the three proposed AI pipelines for intrusion detection. Subsections 4.1 – 4.3 demonstrate the classification performances of the approaches A, B and C, respectively, whereas an overall comparative assessment is performed in Section 4.4.

4.1 Intrusion detection pipeline A

Two variants of pipeline A were proposed in Subsection 3.3.1, in which two different FS algorithms were employed. First of all, the hybrid FS approach was utilized in the proposed pipeline A and a comparative analysis was performed to identify the optimal ML model that maximizes the detection performance. At a second phase, the optimal ML model was selected to be the basis for the sequential backward feature elimination technique (2nd variant) in an attempt to further enhance its classification capacity. The results are presented in the following.

A. Hybrid FS approach

Figure 8 shows the testing classification accuracy of pipeline A with respect to the number of selected features as they have been selected by the hybrid FS approach. The overall best testing performance (82.47%) was obtained by AdaBoost on the first 30 selected features.

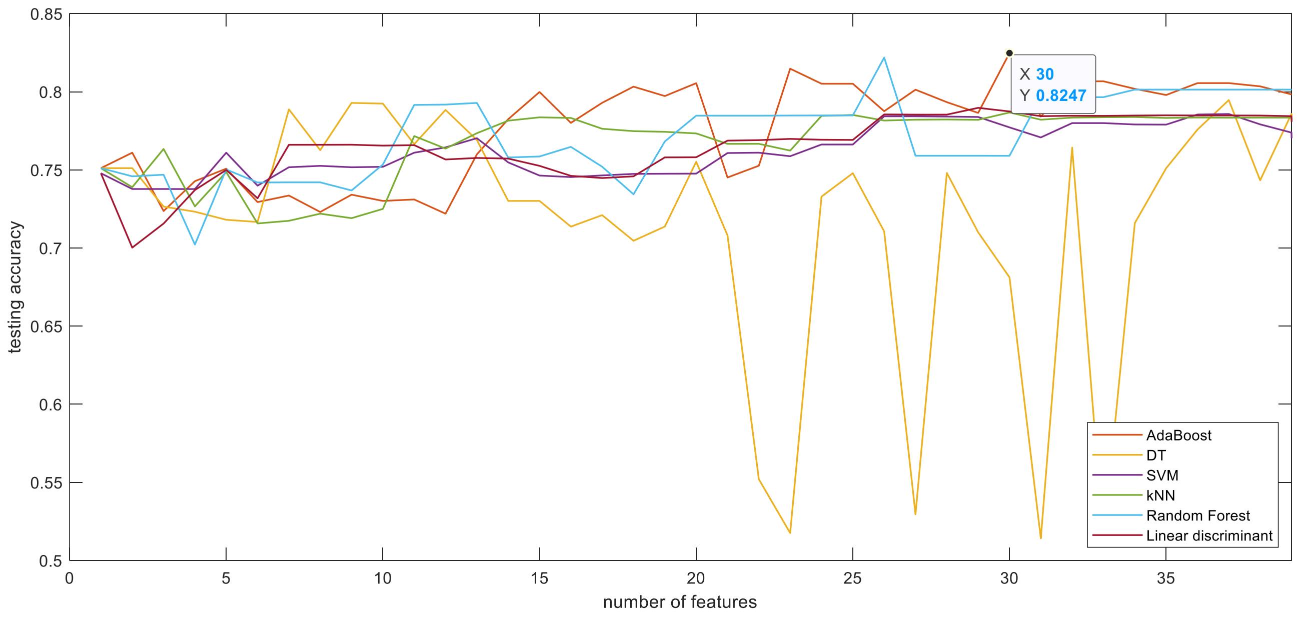


Figure 8 *Testing accuracies with respect to number of selected features (Pipeline A with hybrid FS)*

Table 3 cites: (i) the best testing accuracies achieved by each one the six competing ML models and (ii) the number of selected features where this accuracy was obtained. Overall, AdaBoost was proved to be the most efficient model, whereas RF was the second most effective model (82.19% at 26 features). The rest of the ML models achieved lower testing accuracies (<80%).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

ML model	Best accuracy	Number of features where the best accuracy was achieved
AdaBoost	82.47%	30
Random Forest	79.47%	37
SVM	78.58%	37
kNN	78.67%	30
DT	82.19%	26
LDA	78.97%	29

Table 3 *Testing accuracies achieved by the competing ML models*

Table 4 cites the first more informative features as they have been selected by the proposal FS approach. The full list of selected features has been sent to FINT (leader of the honeypot) in order to be considered in the development of the component.

Feature ID	Feature name	Feature description
34	Dst host same srv rate	% of connections from the same services to the destination host
36	Dst host same src port rate	% of connections from the port services to the destination host
23	Count	number of connections to the same host as the current connection in the past two seconds
3	Service	network service on the destination, e.g., http, telnet, etc.
5	Source bytes	number of data bytes from source to destination
25	Serror rate	number of connections to the same host as the current connection in the past two seconds
10	Hot	number of hot indicators
27	Rerror rate	% of connections that have REJ errors
2	Protocol type	type of the protocol, e.g. tcp, udp, etc
1	Duration	length (number of seconds) of the connection
41	Dst host srv error rate	% of connections that have REJ errors from the same service to the destination host

Table 4 *Most informative features selected by the FS algorithm*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4.2 Intrusion detection pipeline B

4.2.1 Data visualization

Figure 9 depicts indicative images generated by the proposed Vec2im for both intrusion and non-intrusion (normal) classes. White areas correspond to features in which a strong membership to the normal class is observed and vice versa for the black areas. Observing the generated images, there is a clear visual distinction between the two classes, with the normal class (Fig. 3a) being represented by whiter images. Some of the pixels receive high values (close to 1) in images from both classes, however there are some areas on the images that are solely activated in one of the two classes creating distinct colour patterns. The objective of the Siamese neural network that follows is to capture these patterns via their convolutional layer and further convert this information into a more compressed representation (reduced feature space).

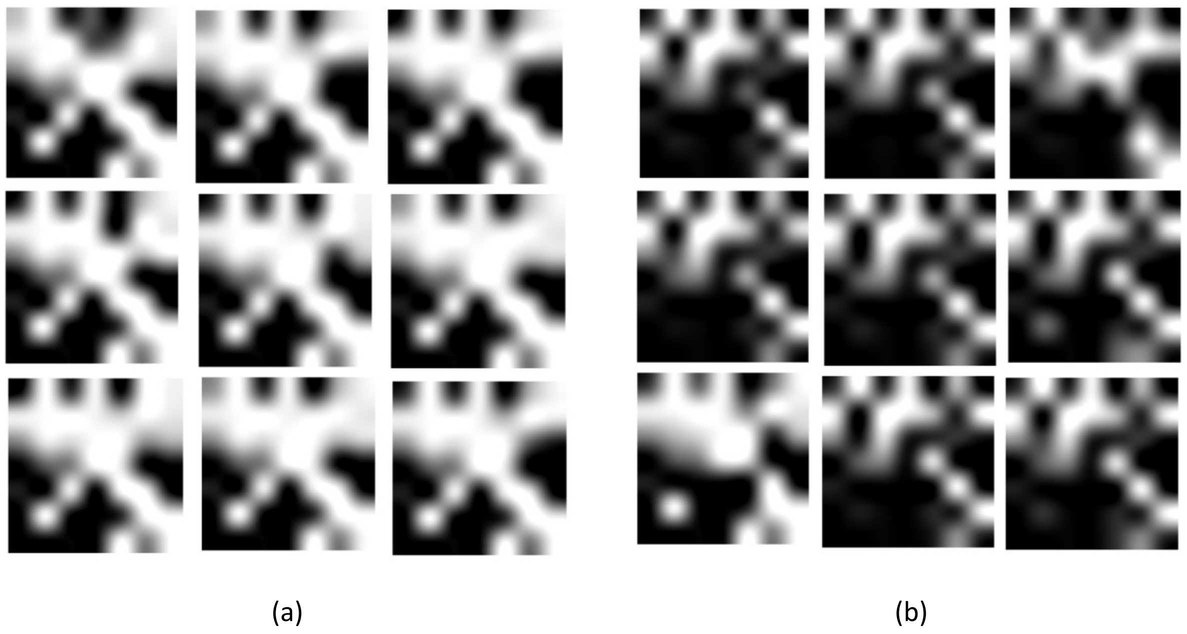


Figure 9 *Indicative images generated by Vec2im for the normal (a) and intrusion class (b)*

4.2.2 Siamese network learning

Figure 10 shows the progression of contrastive loss of the SCNN with respect to the number of iterations. One critical aspect of SCNN training is the termination of the learning process. A validation subset was held out from the training set (a randomly selected 10%) and a linear classifier (LDA in our paper), trained on the 90% of the training set, was utilized to act as a termination criterion. Specifically, the learning process terminates on the iteration where the validation performance of the trained LDA models reaches its maximum value (at iteration 441 in our example). Figures 11 and 12 depict the histogram of the extracted feature values on the reduced 1-d space (that is actually the SCNN output) at iterations 1 and 441 iterations, respectively. The histogram at the first iteration shows that there is a significant overlap between the data distribution of the two classes. On the contrary, the resulted space at iteration 441 is more informative with a small overlap between the per-class distributions (Figure 12) and with most of data points concentrated at the distribution edges.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

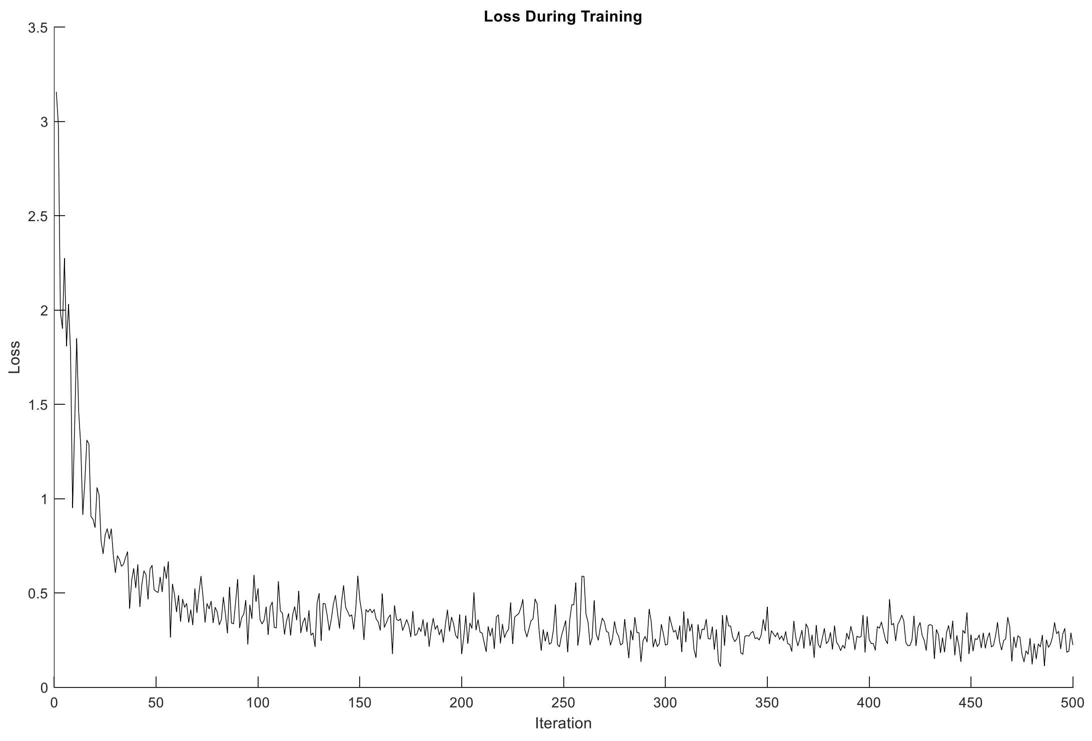


Figure 10 *Training contrastive loss with respect to number of iterations*

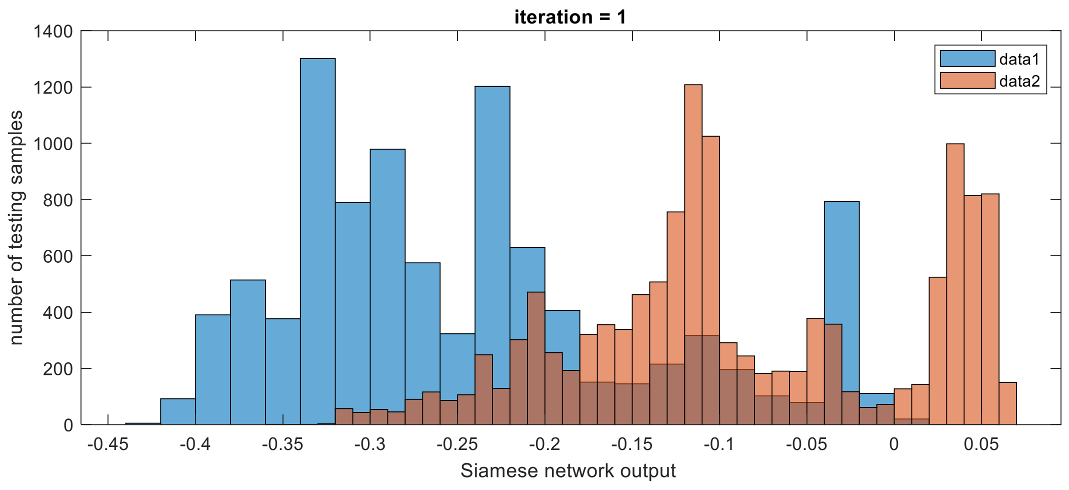


Figure 11 *Histogram of the reduced feature space (SCNN output) at iteration 1*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

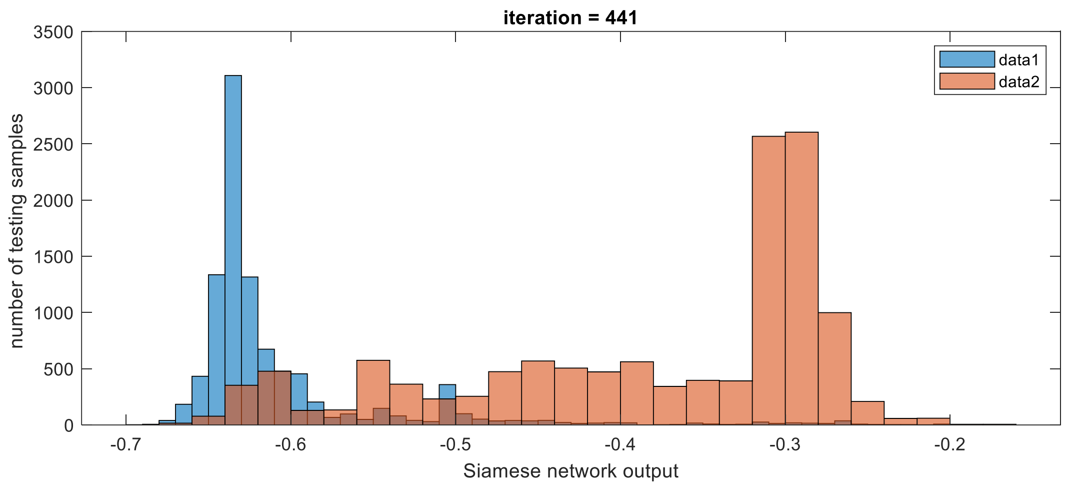


Figure 12 *Histogram of the reduced feature space (SCNN output) at iteration 441*

4.2.3 Comparative analysis

a. Identifying the optimum ML performer on the reduced feature space

Seven ML models were investigated for their suitability on discriminating intrusion from non-intrusion data. Specifically, we tested: (i) Naïve Bayes, (ii) AdaBoost, (iii) Random Forest, (iv) Support Vector Machines (SVM), (v) nearest neighbor classifier (kNN), (vi) decision trees (DT) and (vii) discriminant analysis trained on the reduced 1-d feature space as have been generated by Vec2im-Siam.

Figure 13 shows the progression of the testing accuracy in relation to the number of iterations for the aforementioned ML models. To implement this, we stored the extracted feature values for both training and testing after the end of each iteration of the SCNN learning process. This led to the creation of 500 1-d training and testing sets in which the seven ML were trained and validated, respectively.

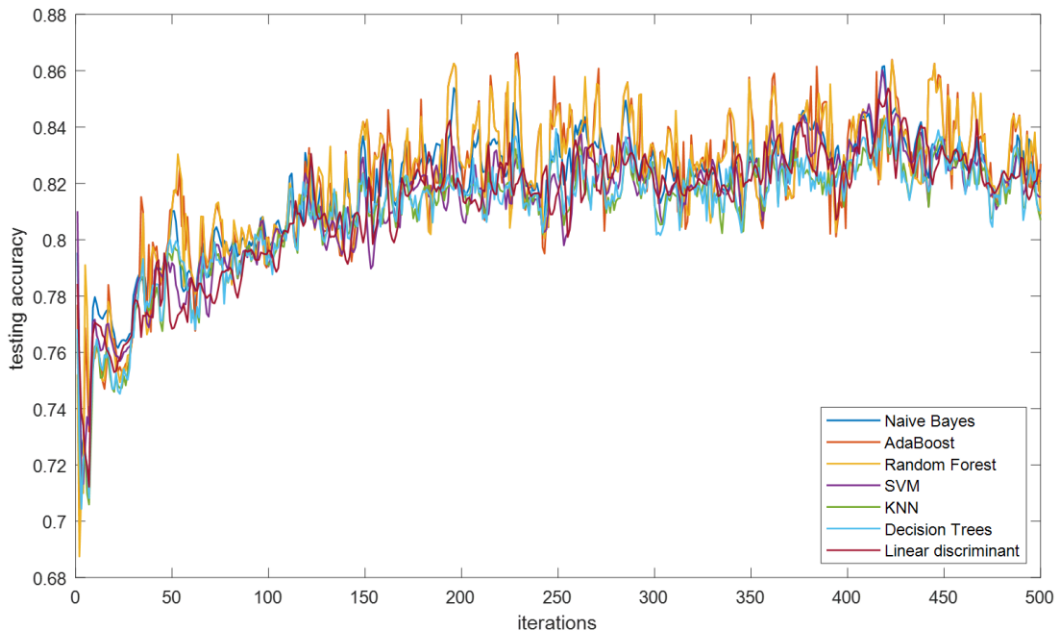


Figure 13 *Testing accuracy with respect to number of iterations of different ML models trained on the resulted 1-d feature space*

Table 5 cites the best performances (training and testing) as accomplished by the seven competing ML models. AdaBoost achieved the overall best testing accuracy (86.64%) whereas slightly reduced testing accuracies (more than 86%) were received by RF, Naïve Bayes and SVM. LDA, DT and kNN were 1% - 2% less effective in our data classification task. Overall, all competing models had similar learning curves (as shown in Figure 13) and they achieved similar testing accuracies within a range of approximately 2%. This finding verifies the effectiveness of the proposed feature extraction methodology that leads to a very informative 1-d feature space in which all ML models (either linear or non-linear) perform well. The confusion matrixes of the four best performing ML models (as shown in Table 6) demonstrate that all four perform similarly exhibiting non important differences in the achieved class accuracies.

Exp.	Feature Extraction	Classification model	Training (%)	Testing (%)
1.	Fuzz-Vec2im -Siam1	Naïve Bayes	98.52	86.17
2.	Fuzz-Vec2im -Siam1	AdaBoost	98.43	**86.64**
3.	Fuzz-Vec2im -Siam1	Random Forest	98.48	86.40
4.	Fuzz-Vec2im -Siam1	SVM	98.53	86.01
5.	Fuzz-Vec2im -Siam1	kNN - 1	**100**	84.29
6.	Fuzz-Vec2im -Siam1	Decision Trees	99.5	84.31
7.	Fuzz-Vec2im -Siam1	Linear Discriminant	98.69	85.38

Table 5 *Comparative analysis with respect to optimal choice of the machine learning model*

		Class 1	Class 2	Per class accuracy
AB	Class 1	9323	388	96.00%
	Class 2	2624	10209	79.55%
			Overall accuracy	**86.64%**

		Class 1	Class 2	Per class accuracy
RF	Class 1	9336	375	96.14%
	Class 2	2692	10141	79.02
			Overall accuracy	**86.40%**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

		Class 1	Class 2	Per class accuracy
NB	Class 1	9393	318	96.73%
	Class 2	2800	10033	78.18%
			Overall accuracy	**86.17%**

		Class 1	Class 2	Per class accuracy
SVM	Class 1	9400	311	96.80%
	Class 2	2842	9991	77.85%
			Overall accuracy	**86.01%**

Table 6 *Confusion Matrixes of the best four performing ML models (LDA and AdaBoost) trained on the reduced feature space*

b. Comparison with other feature selection / feature extraction techniques

A thorough comparative analysis between the proposed methodology and other well-known competing feature extraction / selection techniques is presented below.

- ***Experiments 8 – 9***: *Evaluating the effect of fuzzification on the proposed feature extraction methodology*

In experiments 8 – 9, we evaluated the effect of fuzzification on the performance of the proposed methodology. To accomplish this, we replaced the proposed fuzzification technique with a standard data normalization into the range [0, 1]. Both pre-processing techniques scale the data into the same range, however they have different characteristics:

- data normalization applies a linear transformation on the data rescaling all features to the same range, whereas
- the proposed fuzzification technique transforms raw data into class memberships (the generated fuzzy values declare at what extend a sample belongs to class 1).

The application of the fuzzy allocation scheme prior to feature extraction had a positive effect on both training and testing accuracies. Specifically, replacing the proposed fuzzification scheme with a standard data normalization technique lowered the testing accuracy by 6% (refer to experiment 9 in Table 7).

- ***Experiments 10 – 17***: *Comparing the proposed feature extraction approach with other FS techniques*

For comparison purposes, a wrapper FS technique was also employed to reduce the feature dimensionality of the initial 41-d space. This technique employs a search strategy to look through the space of possible feature subsets, evaluating each subset based on the quality of the performance of a given algorithm. A sequential forward selection strategy was implemented that starts with no feature and progressively adds one feature at a time. The same classifier was utilized in both the wrapper FS and the proposed vec2im+Siam methodology in

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

order to set a fair comparison ground. The wrapper FS technique was implemented four times setting different termination criteria as follows:

- in the experiment 10, the wrapper FS was implemented to identify the most important feature from the entire feature space. The obtained 1-d space was compared with the 1-d space as it has been generated by the proposed in this paper methodology. A much lower testing accuracy was observed in experiment 10 (77.91%) indicating that the selected feature is less informative compared to the extracted feature from Vec2im-Siam.
- In experiments 12, 14 and 16, different termination criteria were set in the Wrapper FS technique leading to feature spaces of higher dimensionality (2-d, 3-d and 10-d, respectively). The discrimination capability of the resulted spaces was also compared with the 1-d space of the proposed methodology using the same classifier. The results verify that the extracted 1-d feature space of Vec2im-Siam is more descriptive than the resulted spaces of the Wrapper FS technique. This is even valid in the case of the 10-d feature space, in which the application of the same classifier led to a testing accuracy of 84.06 (exp.16) that is 2.6% lower than the testing performance of the proposed methodology implemented on a 1-d space.

To further evaluate the usefulness of the proposed fuzzification routine, we applied it as a pre-processing tool prior to the application of the Wrapper FS technique (refer to experiments 11, 13, 15 and 17) and finally compared the classification accuracy obtained with the one obtained in experiments 10, 12, 14 and 16 (where a standard data normalization is employed). Equal accuracies were obtained in experiments 10 and 11 (77.91 % in testing for both) and in experiments 12 and 13 (82.04% in testing for both). A slight increase was observed in the testing accuracy of experiments 15 and 17 (85.81% and 85.93%, respectively) compared to the accuracies achieved in experiments 14 and 16 (82.98% and 84.06%, respectively) indicating that the proposed fuzzification algorithm might also have a positive effect on a variety of other machine learning pipelines.

- ***Experiments 18 – 20****: Comparing the proposed feature extraction approach with PCA*

The proposed Vec2im-Siam feature extraction methodology was finally compared with Principal Component Analysis (PCA) that is a well common feature dimensionality reduction approach. Three different feature spaces of varying dimensionality were generated via PCA as follows:

- A 1-d feature space using only the first extracted principal component (experiment 18)
- A 2-d feature space using the first two extracted principal components (experiment 19)
- A 3-d feature space using the first three extracted principal components (experiment 20).

The results highlighted the superiority of the Vec2im-Siam given that it outperformed PCA by more than 10% in testing even in the case in which higher dimensional spaces were used (e.g. in experiments 19 and 20).

Exp.	Preprocessing	Feature Extraction	Dimensionality of the resulted feature space	Training (%)	Testing (%)
8.	Fuzzification	Vec2im – Siam1	1	**98.69**	**86.64**
9.	Normalization	Vec2im – Siam1	1	90.20	80.64
10.	Normalization	FS (best feature)	1	82.84	77.91
11.	Fuzzification	FS (best feature)	1	82.84	77.91
12.	Normalization	FS (2 first features)	2	90.79	82.04

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

13.	Fuzzification	FS (2 first features)	2	90.79	82.04
14.	Normalization	FS (3 first features)	3	92.20	82.98
15	Fuzzification	FS (3 first features)	3	91.24	85.81
16.	Normalization	FS (10 first features)	10	92.33	84.06
17.	Fuzzification	FS (10 first features)	10	92.03	85.93
18.	Normalization	PCA (1 principal component)	1	89.69	75.64
19.	Normalization	PCA (2 principal components)	2	89.47	76.30
20.	Normalization	PCA (3 principal components)	3	89.48	76.32

Table 7 *Comparative analysis with respect to competing feature selection / extraction techniques using the same classifier (LDA)*

A journal paper presenting the design and the results of the proposed pipeline B was submitted in Springer Open, Cybersecurity[2] journal:

S. Moustakidis, P. Karlsson, A novel feature extraction approach using Siamese convolutional neural networks for intrusion detection, **Cybersecurity**, Springer Open, under review

[2] https://cybersecurity.springeropen.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

4.3 Intrusion detection pipeline C

The results of the unsupervised pipeline C for intrusion detection are presented in this subsection. Pipeline C was applied repeatedly on feature subset of progressively increased dimensionality (as driven by the proposed FS algorithm). The classification accuracy on the testing dataset is depicted in Figure 14 below with respect to the number of features.

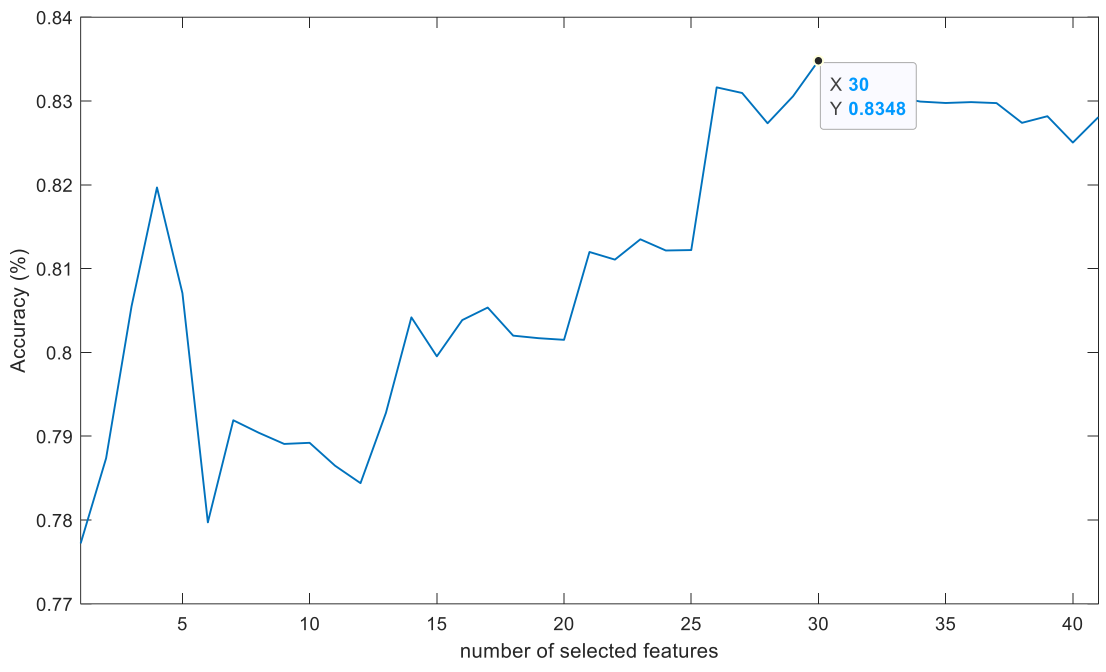


Figure 14 *Testing accuracy with respect to number of features (pipeline C)*

It was observed that the best testing accuracy (83.48%) was achieved at 30 features (similarly to pipeline A), whereas a first peak in the testing accuracy (82%) was obtained using only the first five selected features.

Figure 15 shows the histogram of the autoencoder's output (reconstruction error) trained at the 30 first selected features. It is clearly shown that classes 1 and 2 have different distributions on their reconstruction errors with a relatively small overlapping area between them. So, applying a final classifier on these autoencoder's output leads to an intrusion detection performance of 83.48%.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

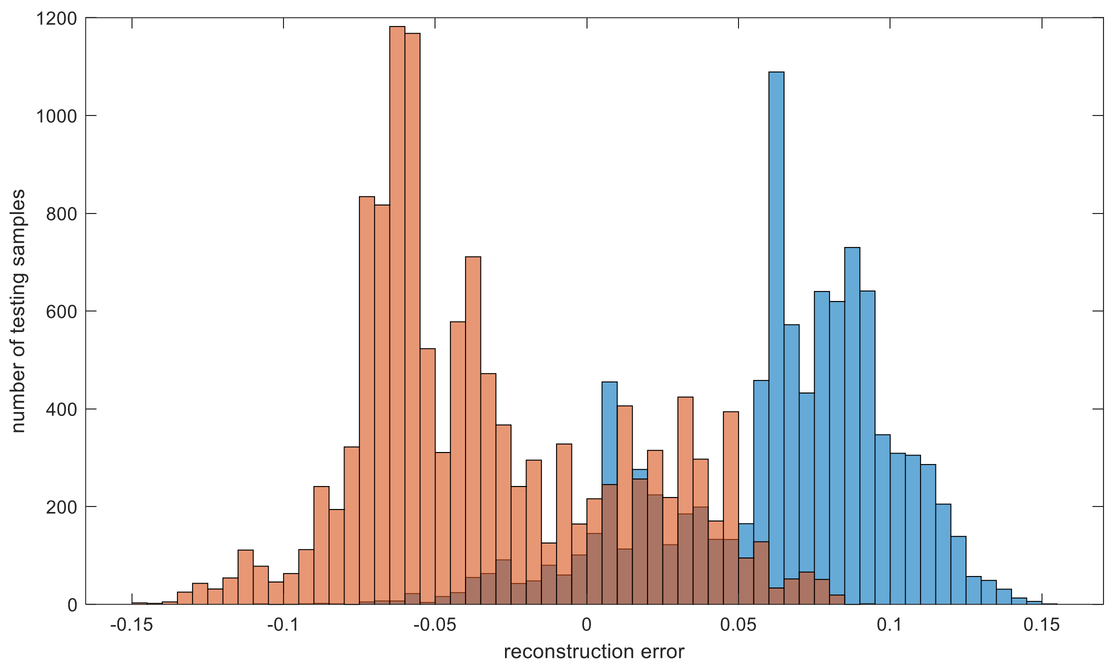


Figure 15 *Histogram of reconstruction error for the testing data point of Class1 (normal – red color) and Class2 (intrusion – blue color)*

4.4 Overall assessment

Table 8 below presents an overall comparative assessment with all three proposed pipelines and their variants. The following conclusions can be drawn from Table 8:

- Pipeline B with AdaBoost performed the best testing accuracy (86.64%). The main advantage of this technique is that it achieves the higher testing accuracy whereas at the same time reduces the feature dimensionality into a 1-D dimensional feature space.
- The unsupervised pipeline C achieved a relatively high testing accuracy (83.48%). It should be stressed out that pipeline C is the one that can be more easily applied in a real case scenario since it only requires normal traffic data (unlabelled).
- Moderate testing accuracies were received by the variants of Pipeline A (76.41% - 82.47%).

Pipeline	Models	Results
Pipeline A *(Supervised with SoA ML)*	Naïve Bayes	76.41%
	AdaBoost	82.47%
	Random Forest	79.47%
	SVM	78.58%
	kNN	78.67%
	Decision Trees	82.19%
	Discriminant	78.97%
Pipeline B *(Supervised with novel DL pipeline)*	Naïve Bayes	86.17%
	AdaBoost	**86.64%**
	Random Forest	86.40%
	SVM	86.01%
	kNN	84.29%
	Decision Trees	84.31%
	Discriminant	85.38%
Pipeline C *(Unsupervised with autoencoder)*	Autoencoder	83.48%

Table 8 *Comparative analysis*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

5 Conclusions and future plans

5.1 Conclusions

This deliverable presented the overall development status of the MLID component on M18 of the project's lifetime and the end of the first interim of MLID two-staged development phases (M10-M18, M22-M30). This is a versioned document and describes the progress of the development first prototype of the component. Within the first development phase of MLID, the following accomplishments have been achieved:

- Feature exploration has been performed and a list of the most informative features (reflecting different aspects of users' behaviour) has been identified
- Three AI pipelines for intrusion detection have been designed, developed and implemented
- Pipeline A employed a standard ML methodology achieving moderate results
- A novel deep learning – based methodology for dimensionality reduction and intrusion detection was designed and tested using Siamese convolutional neural networks (pipeline B)
- A journal paper was prepared and submitted in Springer Open
- An unsupervised pipeline was implemented and tested that requires only normal traffic data to be trained (pipeline C)
- An extensive comparative analysis was performed between all three pipelines and their variants

5.2 Integration plan

The second version of this document will present all the foreseen functionalities of the MLID component and will report the second and final prototype development of the components' development. The integration plan between the MLID component and the rest linked components of SPHINX will be implemented along the following steps.

1. The Honeypot (HP) collects real time information from users' behaviour
2. A feature vector is formulated by the HP comprising traffic features that correspond to a specific pre-determined time period
3. The feature vector is sent to the MLID component
4. The MLID component processes the feature vector and produces a classification decision.
5. The classification output is sent to: (i) the HP and (ii) the DSS that will convert the received information into actionable rules / actions.

The second version of this deliverable will demonstrate the detection capacity of the MLID component on the actual data received by the HP.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Annex I: **References**

[1] S. Sharma and R. Gupta, "Intrusion detection system: A review," *International Journal of Security and Its Applications,* vol. 9, pp. 69-76, 2015.

[2] M. Bijone, "A survey on secure network: intrusion detection & prevention approaches," *American Journal of Information Systems,* vol. 4, pp. 69-88, 2016.

[3] A. Sahasrabuddhe, S. Naikade, A. Ramaswamy, B. Sadliwala, and P. Futane, "Survey on intrusion detection system using data mining techniques," *Int Res J Eng Technol,* vol. 4, pp. 1780-4, 2017.

[4] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials,* vol. 16, pp. 266-282, 2013.

[5] A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, "Using feature selection for intrusion detection system," in *2012 international symposium on communications and information technologies (ISCIT)*, 2012, pp. 296-301.

[6] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*: CRC press, 2016.

[7] N. Kshetri and J. Voas, "Hacking power grids: A current problem," *Computer,* vol. 50, pp. 91-95, 2017.

[8] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?," *IEEE Signal Processing Magazine,* vol. 35, pp. 41-49, 2018.

[9] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & security,* vol. 24, pp. 295-307, 2005.

[10] K. Bajaj and A. Arora, "Dimension reduction in intrusion detection features using discriminative machine learning approach," *International Journal of Computer Science Issues (IJCSI),* vol. 10, p. 324, 2013.

[11] A. Khraisat, I. Gondal, and P. Vamplew, "An anomaly intrusion detection system using C5 decision tree classifier," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2018, pp. 149-155.

[12] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Systems with Applications,* vol. 42, pp. 193-202, 2015.

[13] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *2013 international conference on pattern recognition, informatics and Mobile engineering*, 2013, pp. 294-299.

[14] S. Subramanian, V. B. Srinivasan, and C. Ramasa, "Study on classification algorithms for network intrusion systems," *Journal of Communication and Computer,* vol. 9, pp. 1242-1246, 2012.

[15] S. Potluri, S. Ahmed, and C. Diedrich, "Convolutional neural networks for multi-class intrusion detection system," in *International Conference on Mining Intelligence and Knowledge Exploration*, 2018, pp. 225-238.

[16] B. Zhang, Y. Yu, and J. Li, "Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1-6.

[17] H. Zhang, X. Yu, P. Ren, C. Luo, and G. Min, "Deep adversarial learning in intrusion detection: A data augmentation enhanced framework," *arXiv preprint arXiv:1901.07949,* 2019.

[18] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—a Pearson redundancy based filter," in *Computer recognition systems 2*, ed: Springer, 2007, pp. 242-249.

[19] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *Journal of King Saud University-Computer and Information Sciences,* vol. 29, pp. 462-472, 2017.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

[20] M. Xiong, X. Fang, and J. Zhao, "Biomarker identification by feature wrappers," *Genome Research,* vol. 11, pp. 1878-1887, 2001.

[21] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint ℓ2, 1-norms minimization," in *Advances in neural information processing systems*, 2010, pp. 1813-1821.

[22] Q. Zhou, H. Zhou, and T. Li, "Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features," *Knowledge-based systems,* vol. 95, pp. 1-11, 2016.

[23] E. Al Daoud, "Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset," *International Journal of Computer and Information Engineering,* vol. 13, pp. 6-10, 2019.

[24] L. Torlay, M. Perrone-Bertolotti, E. Thomas, and M. Baciu, "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy," *Brain informatics,* vol. 4, pp. 159-169, 2017.

[25] L. Fraiwan, K. Lweesy, N. Khasawneh, H. Wenz, and H. Dickhaus, "Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier," *Computer methods and programs in biomedicine,* vol. 108, pp. 10-19, 2012.

[26] T. Kobayashi, T. Kannari, H. Horiuchi, N. Matsui, T. Ito, K. Nojin*, et al.*, "Predictors affecting balance performances in patients with knee osteoarthritis using decision tree analysis," *Osteoarthritis and Cartilage,* vol. 27, p. S243, 2019.

[27] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," *Procedia Technology,* vol. 4, pp. 119-128, 2012.

[28] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," *Pattern Recognition,* vol. 45, pp. 1318-1325, 2012.

[29] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based $ k $-nearest-neighbor for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 48, pp. 4099-4109, 2010.

[30] Y. Qian, M. Ye, and J. Zhou, "Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 51, pp. 2276-2291, 2012.

[31] S. K. Lodha, D. M. Fitzpatrick, and D. P. Helmbold, "Aerial lidar data classification using adaboost," in *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*, 2007, pp. 435-442.

[32] K. Torkkola, "Linear discriminant analysis in document classification," in *IEEE ICDM Workshop on Text Mining*, 2001, pp. 800-806.

[33] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," presented at the Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01, 2005.

[34] K. Chen and A. Salman, "Extracting speaker-specific information with a regularized siamese deep network," in *Advances in Neural Information Processing Systems*, 2011, pp. 298-306.

[35] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *Proceedings of the fifteenth conference on computational natural language learning*, 2011, pp. 247-256.

[36] E. Lei, O. Castañeda, O. Tirkkonen, T. Goldstein, and C. Studer, "Siamese neural networks for wireless positioning and channel charting," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2019, pp. 200-207.

[37] A. Mehmood, M. Maqsood, M. Bashir, and Y. Shuyuan, "A Deep Siamese Convolution Neural Network for Multi-Class Classification of Alzheimer Disease," *Brain Sciences,* vol. 10, p. 84, 2020.

[38] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, "Siamese convolutional neural networks for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters,* vol. 16, pp. 1200-1204, 2019.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

[40] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks," *chapter Convolutional networks for images, speech, and time series,* pp. 255-258, 1998.

[41] F. Wang, L. Kang, and Y. Li, "Sketch-based 3d shape retrieval using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1875-1883.

[42] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[43] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 665-674.

[44] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Icml*, 2011.

[45] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.,* vol. 11, pp. 3371-3408, 2010.

[46] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE,* vol. 2, pp. 1-18, 2015.

[47] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737-744.

[48] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, 2015.

[49] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, pp. 1735-1742.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[51] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification, volume November. John Wily & Sons," *Inc., New York,* vol. 2, 2000.

[52] W. A. Belson, "Matching and Prediction on the Principle of Biological Classification," *Journal of the Royal Statistical Society. Series C (Applied Statistics),* vol. 8, pp. 65-75, 1959.

[53] I. H. Witten, E. Frank, and M. Hall, "a.(2011). Data Mining: Practical Machine Learning Tools and Techniques," *Annals of Physics,* vol. 54, 2011.

[54] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, September 01 1995.

[55] B. Scholkopf, "Support vector learning," *Ph. D. thesis, Technische Universitat Berlin.,* 1997.

[56] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences,* vol. 55, pp. 119-139, 1997.

[57] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, pp. 5-32, October 01 2001.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Annex II: **Submitted paper**

Cybersecurity

A novel feature extraction approach using Siamese convolutional neural networks for intrusion detection

--Manuscript Draft--

Manuscript Number:	CYSE-D-20-00059
Full Title:	A novel feature extraction approach using Siamese convolutional neural networks for intrusion detection
Article Type:	Research
Funding Information:	H2020 Societal Challenges (826183)
Abstract:	Intrusion detection systems (IDS) can play a significant role in detecting security threats or malicious attacks that aim to steal information and/or corrupt network protocols. To deal with the dynamic and complex nature of cyber-attacks, advanced intelligent tools have been applied resulting into powerful and automated IDS that rely on the latest advances of machine learning (ML) and deep learning (DL). Most of the reported effort has been devoted on building complex ML/DL architectures adopting a brute force approach towards the maximization of their detection capacity. However, just a limited number of studies have focused on the extraction of user-friendly risk indicators that could be easily used by security experts. Many papers have explored various dimensionality reduction algorithms, however a large number of selected features is still required to detect the attacks successfully. The objective of this paper is to transform the available data collected by IDS into a single actionable and easy-to-understand risk indicator. To achieve this a novel feature extraction pipeline was implemented consisting of the following components: (i) a fuzzy allocation scheme that transforms raw data to fuzzy class memberships, (ii) a mechanism for converting feature vectors to images (Vec2im) and (iii) a dimensionality reduction module that makes use of Siamese convolutional neural networks that finally reduces the input data dimensionality into a 1-d feature space. The performance of the proposed methodology was validated via a thorough comparative analysis that demonstrated its effectiveness over a number of well-known feature selection (FS) and extraction techniques. The output of the proposed feature extraction pipeline could be potentially used by security experts as an indicator of malicious activity, whereas the generated images could be further utilized and/or integrated as a visual analytics tool in existing IDS.
Corresponding Author:	Serafeim Moustakidis
Corresponding Author E-Mail:	s.moustakidis@aideas.eu
Corresponding Author's Institution:	AiDEAS OU
First Author:	Serafeim Moustakidis

Order of Authors:	Serafeim Moustakidis Patrik Karlsson

Title

A novel feature extraction approach using Siamese convolutional neural networks for intrusion detection

Authors:

Serafeim Moustakidis[1,*], Patrik Karlsson[1]

1. AIDEAS OÜ, Narva mnt 5, Tallinn, Harju maakond, Estonia

* Correspondence: s.moustakidis@aideas.eu

Abstract

Intrusion detection systems (IDS) can play a significant role in detecting security threats or malicious attacks that aim to steal information and/or corrupt network protocols. To deal with the dynamic and complex nature of cyber-attacks, advanced intelligent tools have been applied resulting into powerful and automated IDS that rely on the latest advances of machine learning (ML) and deep learning (DL). Most of the reported effort has been devoted on building complex ML/DL architectures adopting a brute force approach towards the maximization of their detection capacity. However, just a limited number of studies have focused on the extraction of user-friendly risk indicators that could be easily used by security experts. Many papers have explored various dimensionality reduction algorithms, however a large number of selected features is still required to detect the attacks successfully. The objective of this paper is to transform the available data collected by IDS into a single actionable and easy-to-understand risk indicator. To achieve this a novel feature extraction pipeline was implemented consisting of the following components: (i) a fuzzy allocation scheme that transforms raw data to fuzzy class memberships, (ii) a mechanism for converting feature vectors to images (Vec2im) and (iii) a dimensionality reduction module that makes use of Siamese convolutional neural networks that finally reduces the input data dimensionality into a 1-d feature space. The performance of the proposed methodology was validated via a thorough comparative analysis that demonstrated its effectiveness over a number of well-known feature selection (FS) and extraction techniques. The output of the proposed feature extraction pipeline could be potentially used by security experts as an indicator of malicious activity, whereas the generated images could be further utilized and/or integrated as a visual analytics tool in existing IDS.

Keywords: feature extraction, Siamese convolutional neural networks, machine learning, intrusion detection

1. Introduction

An IDS is a security tool that collects information from various sources (e.g. routers, computers, network data) aiming at identifying malicious activities and/or users that attempt to either get access to computers, steal protected data or even manipulate and disable information systems (Sharma and Gupta 2015). IDSs can be categorized into three main categories (Biijone 2016). The first category of IDS compares the collected patterns of network traffic with specific and pre-determined signatures (attack patterns). An attack is detected once there is match with an already known pattern, however this kind of IDS is incapable of identifying new (unknown) malicious activities. The second category builds on a set of rules and thresholds (specifications) that have been manually specified by security experts. These specification-based IDSs do not generate false alarms when unusual (but legitimate) program behaviors are encountered but in general the specifications development is a tedious and expensive process while the specified set of rules is often very difficult to evaluate and verify. Unlike signature and specification IDSs, automated intrusion detection (AID) systems is a new category that employs machine learning, statistical-based or knowledge-based methods to define a normal model of the behavior of a computer system. The effectiveness of AID systems depends a lot on the quantity as well as quality of the network traffic patterns that are used as data instances during their training.

In the last few decades, ML has been used to improve intrusion detection (Sahasrabuddhe et al. 2017). There is a large number of related studies using various synthetic datasets (such as KDD-Cup 99 or DARPA 1999 datasets) to develop and validate ML-empowered AID systems. Any significant deviation between the observed 'normal' behavior can be regarded as an anomaly, which can be then interpreted as an intrusion. The main assumption of the aforementioned approaches is that malicious behavior differs from typical user behavior. One simplistic method to decide whether a behavior is normal or abnormal is by comparing it with the standard deviation of the normal user behaviors in the training dataset. Any example exceeding the pre-determined threshold (e.g. three times the standard deviation) could be classified in the intrusion category. ML provides a more sophisticated method for decision making overcoming the deficiencies of the heuristic approaches (such the manual selection of the threshold etc.). Development of ML-based AID systems comprises of two phases: the training phase and the testing phase.

c. In the training phase, the normal traffic profile is used to learn a model of normal behavior,
d. In the testing phase, a new data set is used to validate the system's capacity to generalize to previously unseen intrusions.

AIDS can be classified into a number of categories based on the method used for training, for instance, statistical based, knowledge-based and machine learning based (Butun et al. 2014). The main advantages of ML-empowered AID systems are: (i) Their ability to identify zero-day attacks without relying on a signature database (Alazab et al. 2012). A danger signal can be triggered when the examined behavior differs from the usual behavior. (ii) Their capability to discover internal malicious activities. An alarm will be created in cases where an intruder starts making transactions in a compromised account that deviate from the typical user activity. (iii)The normal user behavior is hidden to intruders and thus it becomes more difficult for them to remain undetected. The objective of using machine learning techniques is to create IDS with improved accuracy and less requirement for prior human knowledge. However, one of the main challenges of current AIDS is the high false positive rates because anomalies may just be new normal activities rather than genuine intrusions.

One of the crucial phases in today's ML pipelines is the process of extracting knowledge from large quantities of data. To effective extract knowledge from raw data, ML relies on a set of rules, methods, or complex "transfer functions" that are applied to find interesting data patterns, or to recognize and predict behavior (Dua and Du 2016). Many ML algorithms (such as clustering, neural networks, association rules, decision trees, genetic algorithms, and nearest neighbor methods) have been recently applied in the area of AIDs for discovering knowledge from intrusion datasets (Kshetri and Voas 2017, Xiao et al. 2018). Some prior research in data mining has examined the use of different algorithms to extract meaningful information for intrusion data. Two feature selection algorithms were investigated by Chebrolu et al. 2015 employing Bayesian networks (BN) and Classification Regression Trees (CRC). The outputs of the aforementioned algorithms were finally combined to increase accuracy. Bajaj et al. 2013 proposed a technique for feature selection using a hybrid approach that combines Information Gain (IG) and correlation attribute evaluation. To validate the discrimination capacity of the selected features, the authors applied several classification algorithms such as C4.5, naïve Bayes, NB-Tree and Multi-Layer Perceptron (Khraisat 2018). Genetic-fuzzy rule mining has been also explored to evaluate the importance of IDS features by Elhag et al. 2015. Thaseen et al. 2012 proposed a Random Tree model to improve

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

the accuracy and reduce the false alarm rate, whereas Subramanian et al. also studied the performance of decision tree algorithms on the NSL-KDD dataset.

Unlike ML approaches that require the extraction of features, Deep learning (DL)-based detection methods learn feature automatically in an end-to-end fashion (directly from raw data to decisions). DL is gradually attracting more interest in AID studies. A CNN-based AID methodology was presented by Potluri et al. 2018 conducting experiments on the NSL-KDD and the UNSW-NB datasets. In the pre-processing phase, the features of the datasets were transformed into images of 8*8 pixels. Then, a three-layer CNN was trained to classify the attacks. Pre-trained deep networks (ResNet 50 and GoogLeNet) were also explored as alternative solutions to the task of extracting new informative features. The proposed CNN performed best, reaching accuracies of 91.14% on the NSL-KDD and 94.9% on the UNSW-NB 15. A sparse autoencoder was also proposed by Zhang et al. 2018 to extract features from the NSL-KDD dataset. The extracted features were supplied to an XGBoost model with the objective to detect attacks. To overcome the observed data imbalance problem, data resampling was employed (using SMOTE). The SMOTE algorithm oversamples the minority classes and divides the majority classes into many subclasses so that every class is balanced. Data augmentation with GANs has been also explored by Zhang et al. 2019. The GAN model was used to generate data similar to the flow data of KDD99. Adding this generated data to the training set increased the generalization capacity of the detection model that was able to identify not only attacks but attack variants as well.

This paper contributes to current AID systems by transforming the available multi-dimensional network traffic data into an actionable and easy-to-understand indicator for security experts. This has been achieved by designing a novel feature extraction pipeline that builds on the latest advances of data mining and ML/DL. Specifically, a fuzzy allocation scheme is initially applied to transform raw data to fuzzy class memberships. Then a data transformation mechanism converts feature vectors to images (Vec2im) and a dimensionality reduction module makes use of Siamese convolutional neural networks to reduce the input data dimensionality into a 1-d feature space. The performance of the proposed methodology was validated via a thorough comparative analysis that demonstrated its effectiveness over a number of well-known feature selection and extraction techniques.

The rest of this article is structured as follows: The proposed feature extraction pipeline with architectural and implementation details is provided in Section 3. Experimental results are demonstrated in Section 4, with a comparative analysis with existing approaches. Conclusions are drawn in Section 5.

2. Methodology

The proposed methodology in this paper for feature extraction in the domain of intrusion detection includes four processing steps: (i) data pre-processing making use of a fuzzy allocation scheme to convert raw data into fuzzy values, (ii) a data transformation technique that generates images comprising of fuzzy memberships, (iii) a novel feature extraction algorithm employing Siamese convolutional neural networks and finally (iv) a learning process for training, and evaluation of the results, as illustrated in Fig. 1. The proposed methodology is thoroughly presented in the following sections.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

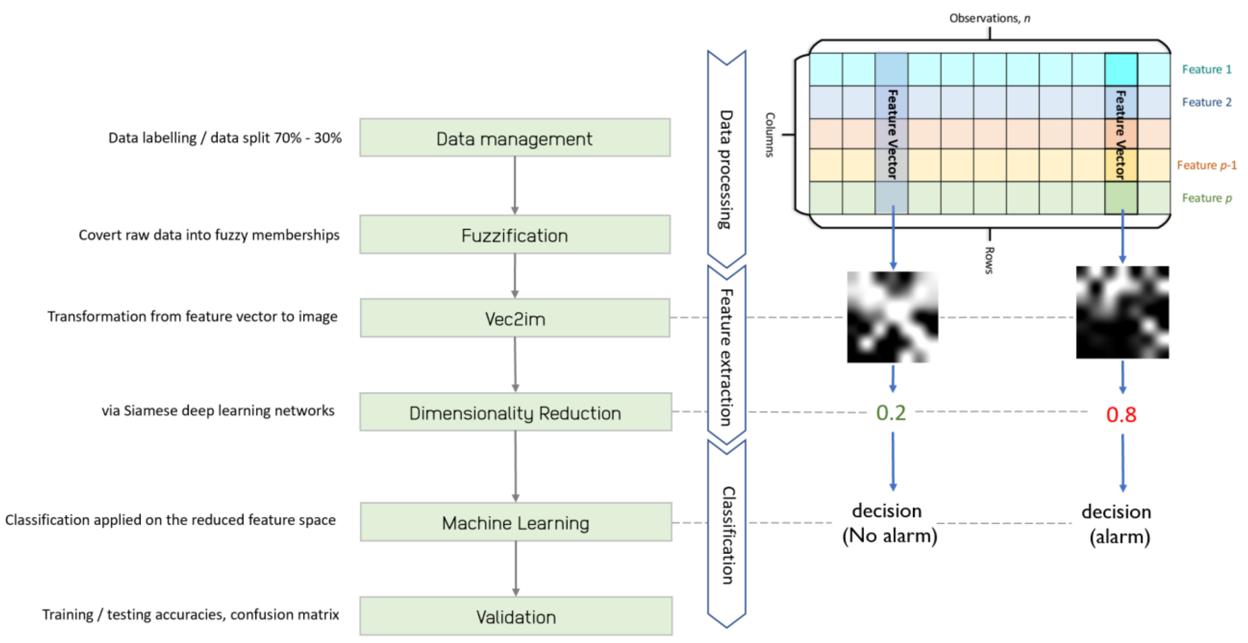


Figure 1 The proposed Vec2im-Siam methodology

2.1 Dataset description

To validate the performance of the proposed feature extraction methodology, the NSL-KDD dataset was employed. NSL-KDD is actually a variant of the KDD99 dataset that is the most widespread IDS benchmark dataset at present. Overcoming the limitations of KDD99 (severely unbalanced data, many duplicated and redundant records), NSL-KDD represents a more balanced dataset with a moderate number of records that has allowed the application of various IDS in a large number of papers whose results are consistent as well as comparable. For the aforementioned reasons, we used it as a benchmark in our analysis.

Our dataset consists of 41 features that are categorized in four subsets, i.e., basic features, content features, host-based statistical features, and time-based statistical features. As far as the attack types in the NSL-KDD dataset, they are divided into four categories:

1. **Denial of service** (DoS): exhausting the resources of the attacked object by savage means; thus, making it unable to provide normal services; paralysis. Subcategories: ping of Death, LAND, neptune, backscatter, smurf, teardrop
2. **R2L**: unauthorized access to remote computers. Subcategories: ftp-write, password guessing, imap, multi-hop, phf, spy, warezclient, warezmaster
3. **U2R**: unauthorized access to local superuser privileges. Subcategories: buffer Overflow, loadmodule, perl, rootkit
4. **Probe**: monitoring and other detection behavior. Subcategories: ipsweeping, nmap, portsweeping, satan

Specifically, a 20% subset of the NSL-KDD training data (the NSL-KDD Train 20 variant) was used in our paper comprising of 25,192 data points, where the NSL-KDD Test+ data file (comprising of 22,544 data points) was utilized for testing. Given that the focus of the paper is not on the recognition of the different attacks, the intrusion detection problem was considered as binary by merging all the anomalous records (categories 1-4) into one class.

2.2 Fuzzification and image formulation (Vec2im)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

First of all, the non-numeric attributes of the dataset were converted into numeric values. For the efficient training of machine learning algorithms, input data is typically transformed by a number of pre-processing routines with data normalization being the gold standard. Different algorithms could be used to normalize the input data (such as min-max normalization or normalization with respect to standard deviation), however in this paper we employed a fuzzy allocation scheme as described below.

Fuzzification: To normalize as well as evaluate the classification capabilities of each feature, we applied a simple fuzzy allocation scheme that assigs varying degrees of patterns to every class. For feature j, the fuzzy membership $u_i(x_{k,j}) \in [0,1]$ indicating the degree to which $x_{k,j}$ belongs to class i is determined by:

$$u_i(x_{k,j}) = \frac{1}{\sum_{m=1}^{c}\left[\frac{(x_{k,j}-u_{i,j})^2}{(x_{k,j}-u_{m,j})^2}\right]^{1/(b-1)}} \qquad (1)$$

where $u_{i,j} = \sum_{k\in A_i} x_{k,j}/N_i$ is the class i mean along the $x_{k,j}$ component, A_i is the set of indexes of the training examples belonging to class i , N_i is the number of class i patterns and b is a fuzzification factor (b = 2 in our experiments). In our paper, every feature component of $x_{k,j}$, was converted to $u_1(x_{k,j})$ that denotes its fuzzy membership to class 1 (normal / non intrusion class). High values of $u_1(x_{k,j})$ close to 1 indicate a strong membership to the non-intrusion class whereas low $u_1(x_{k,j})$ values close to 0 are representative of examples belonging to the malicious class.

Vec2im: At the second phase of processing, the generated memberships were re-placed in a matrix format resulting to one grey-scale image per example. Specifically, the 41 features $x_{k,j}, j = 1,\dots,41$ were transformed to 41 fuzzy memberships $u_i(x_{k,j}), j = 1,\dots,41$ and finally a 7×7 image was created per sample by placing the fuzzy memberships in a matrix as presented in Figure 2. Zero values were also included in the matrix in random cells to fill the eight gaps (given that the dimensionality of the initial feature set was 41 with a total of 49 cells to be filled in the matrix). Fuzzy memberships and zero values were ordered randomly since it was concluded that their order has not any significant impact on the final performance of the proposed methodology.

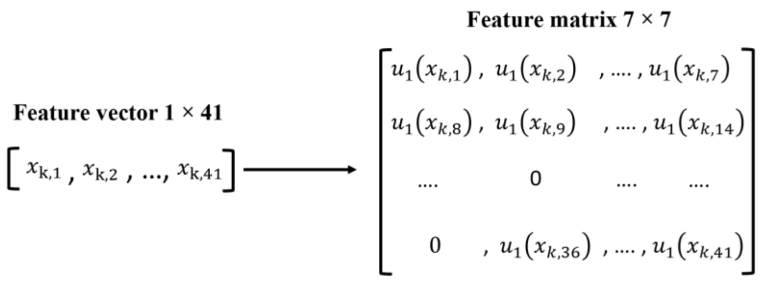

Feature matrix 7 × 7
Feature vector 1 × 41
$[x_{k,1}, x_{k,2}, \dots, x_{k,41}]$
$u_1(x_{k,1}), u_1(x_{k,2}), \dots, u_1(x_{k,7})$
$u_1(x_{k,8}), u_1(x_{k,9}), \dots, u_1(x_{k,14})$
.... 0 ....
0, $u_1(x_{k,36}), \dots, u_1(x_{k,41})$

Figure 2. Converting feature vectors to images (Vec2im)

2.3 Dimensionality Reduction with Siamese deep learning networks

Deep Siamese Convolutional Neural Networks (SCNN) architecture is a variant of neural networks that was originally designed to solve signature verification problem of image matching (Bromley et al. 1993). It has also been used for one-shot image classification (Koch 2015), face verification where the categories are not known in advance (Chopra et al. 2005) as well as for dimensionality reduction (Hadsell et al. 2006). SCNN consist of two identical symmetric CNN subnetworks that share the same weights. In our experiment, each identical CNN was built using one convolutional layer followed by three fully connected layers. The rectified linear units (ReLU) nonlinearity was applied as the activation function for all layers, and adaptive moment estimation (ADAM) optimizer was utilized to control learning rate (Kingma and Ba 2014). The similarity between images was calculated by Euclidean distance, and the contrastive loss (Chopra et al. 2005) was calculated to define the loss function as follows:

$$\mathcal{L}(W, I_1, I_2) = \mathbf{1}(L=0)\frac{1}{2}D^2 + \mathbf{1}(L=1)\frac{1}{2}[\max(0, margin - D)]^2 \tag{2}$$

$$\text{where } D = \|f(I_1) - f(I_2)\|^2, \tag{3}$$

I_1 and I_2 are a pair of the generated images fed into each of two identical CNNs. $1(\cdot)$ is an indicator function to show that whether two images have the same label, where L = 0 represents the images have the same label and L = 1 represents the opposite. W is the shared parameter vector comprising of the weights that both neural networks share each other. $f(I_1)$ and $f(I_2)$ are the latent representation vectors of input I_1 and I_2, respectively and D is the Euclidean distance between them. The selected SCNN architecture (as depicted in Figure 3) reduces the dimensionality of the 41-dimensional feature space to a single 1-d space.

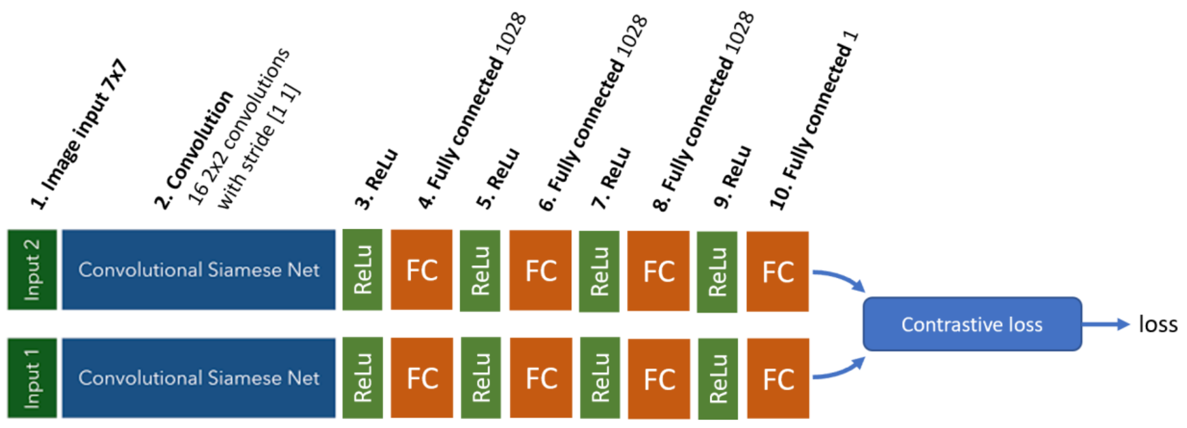


Figure 3 Selected Siamese network architecture for dimensionality reduction

2.4 Decision making on the reduced feature space

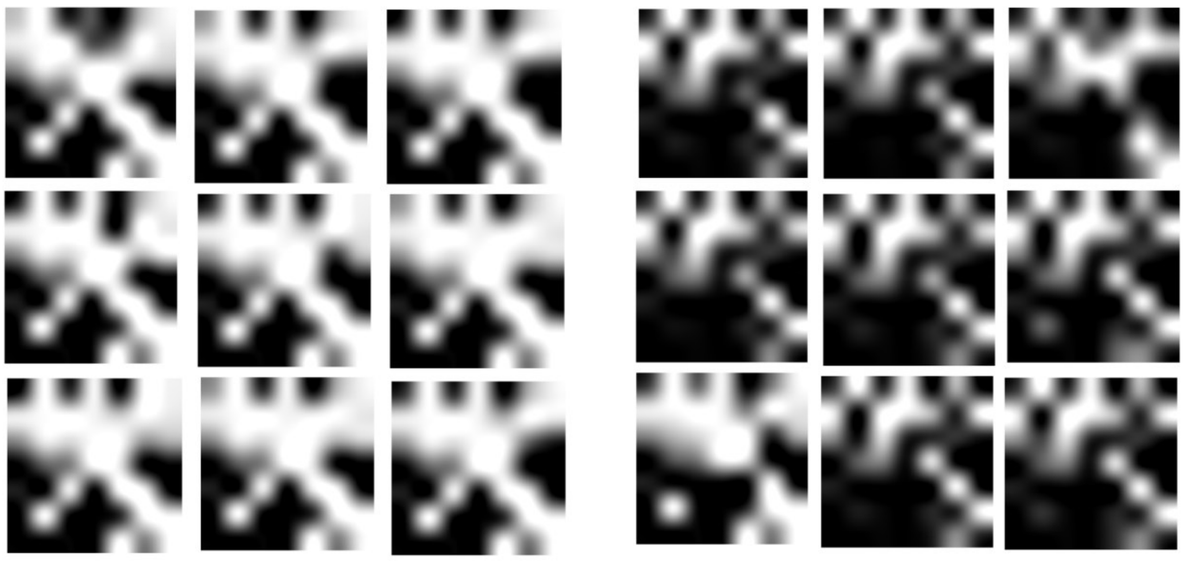
To evaluate the discrimination capacity of the extracted features, we employed various machine learning models trained to implement the binary classification task on the resulted 1-d space. We tested linear discriminant analysis (LDA) and Naïve Bayes (Duda et al. 2000) to provide a baseline for comparisons with more advanced models. We also evaluated decision trees (Belson 1959, Witten et al. 2011), driven by Gini's diversity index, KNN (Atekson et al. 1997), as well as non-linear support vector machines (SVM) algorithms (Cortes and Vapnik 1995, Scholkopf 1997) with Gaussian kernel, which can deal with the overfitting problems that appear in high-dimensional spaces. The ensemble techniques AdaBoost (Freund and Schapire 1997) and Random Forest (Breiman 2001) were also evaluated using DT models as weak learners.

To achieve a fair comparison between the different approaches, hyperparameter selection was performed for each one of the investigated machine algorithms. A validation subset was held out from the training set (a randomly selected 10%) as a criterion for: (i) selecting the optimum hyperparameters by means of a grid search process as well as (ii) deciding the termination of the SCNN learning.

3. Results

3.1 Data visualization

Figure 4 depicts indicative images generated by the proposed Vec2im for both intrusion and non-intrusion (normal) classes. White areas correspond to features in which a strong membership to the normal class is observed and vice versa for the black areas. Observing the generated images, there is a clear visual distinction between the two classes, with the normal class (Fig. 3a) being represented by whiter images. Some of the pixels receive high values (close to 1) in images from both classes, however there are some areas on the images that are solely activated in one of the two classes creating distinct color patterns. The objective of the Siamese neural network that follows is to capture these patterns via their convolutional layer and further convert this information into a more compressed representation (reduced feature space).



(a) (b)

Figure 4 Indicative images generated by Vec2im for the normal (a) and intrusion class (b)

3.2 Siamese network learning

Figure 5a shows the progression of contrastive loss of the SCNN with respect to the number of iterations. One critical aspect of SCNN training is the termination of the learning process. A validation subset was held out from the training set (a randomly selected 10%) and a linear classifier (LDA in our paper), trained on the 90% of the training set, was utilized to act as a termination criterion. Specifically, the learning process terminates on the iteration where the validation performance of the trained LDA models reaches its maximum value (at iteration 441 in our example). Figures 5b and 5c depict the histogram of the extracted feature values on the reduced 1-d space (that is actually the SCNN output) at iterations 1 and 441 iterations, respectively. The histogram at the first iteration shows that there is a significant overlap between the data distribution of the two classes. On the contrary, the resulted space at iteration 441 is more informative with a small overlap between the per-class distributions (Figure 5c) and with most of data points concentrated at the distribution edges.

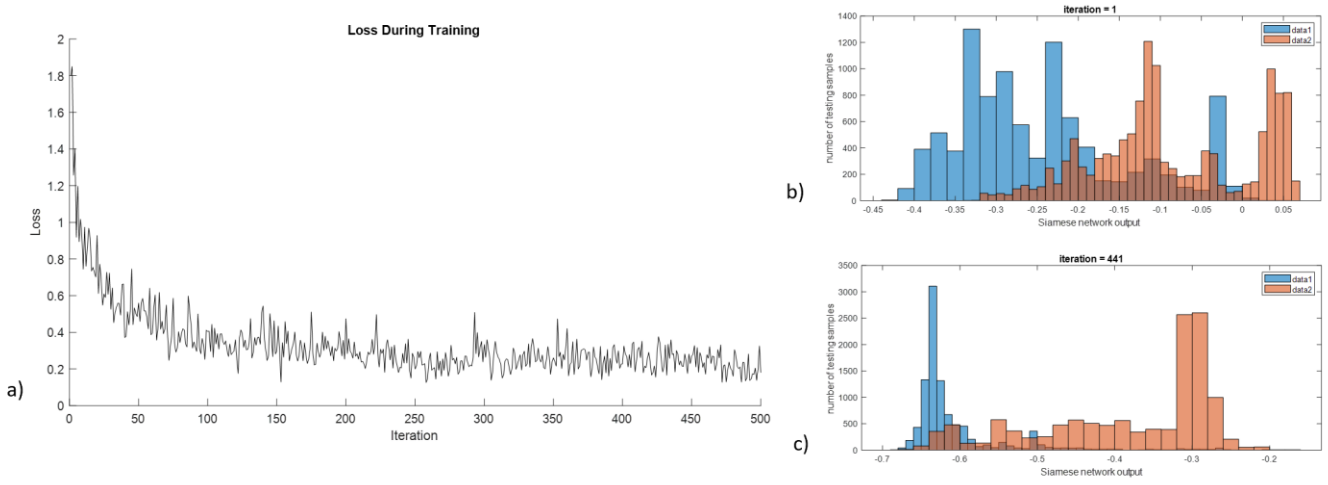


Figure 5a) Training contrastive loss with respect to number of iterations, b) histogram of the reduced feature space (SCNN output) at iteration 1 and c) histogram of the reduced feature space (SCNN output) at iteration 441

3.3 Comparative analysis

a. Identifying the optimum ML performer on the reduced feature space

Seven ML models were investigated for their suitability on discriminating intrusion from non-intrusion data. Specifically, we tested: (i) Naïve Bayes, (ii) AdaBoost, (iii) Random Forest, (iv) Support Vector Machines (SVM), (v) nearest neighbor classifier (kNN), (vi) decision trees (DT) and (vii) discriminant analysis trained on the reduced 1-d feature space as have been generated by Vec2im-Siam.

Figure 6 shows the progression of the testing accuracy in relation to the number of iterations for the aforementioned ML models. To implement this, we stored the extracted feature values for both training and testing after the end of each iteration of the SCNN learning process. This led to the creation of 500 1-d training and testing sets in which the seven ML were trained and validated, respectively.

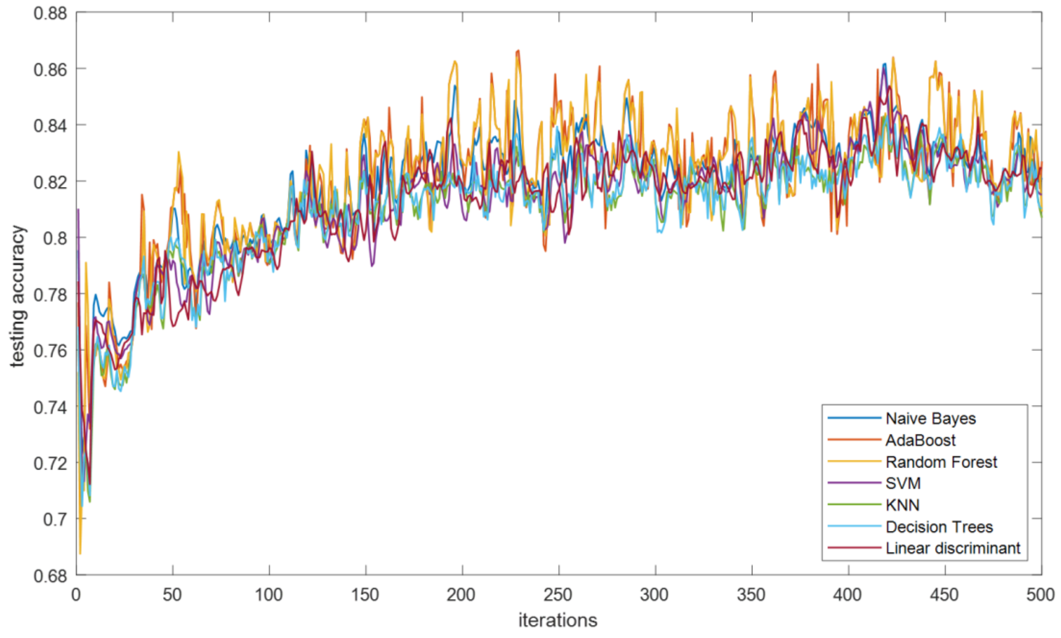


Figure 6 Testing accuracy with respect to number of iteration of different ML models trained on the resulted 1-d feature space

Table 1 cites the best performances (training and testing) as accomplished by the seven competing ML models. AdaBoost achieved the overall best testing accuracy (86.64%) whereas slightly reduced testing accuracies (more than 86%) were received by RF, Naïve Bayes and SVM. LDA, DT and kNN were 1% - 2% less effective in our data classification task. Overall, all competing models had similar learning curves (as shown in Figure 6) and they achieved similar testing accuracies within a range of approximately 2%. This finding verifies the effectiveness of the proposed feature extraction methodology that leads to a very informative 1-d feature space in which all ML models (either linear or non-linear) perform well. The confusion matrixes of the four best performing ML models (as shown in Table 2) demonstrate that all four perform similarly exhibiting non important differences in the achieved class accuracies.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

Table 1 Comparative analysis with respect to optimal choice of the machine learning model

Exp.	Feature Extraction	Classification model	Training (%)	Testing (%)
1.	Fuzz-Vec2im -Siam1	Naïve Bayes	98.52	86.17
2.	Fuzz-Vec2im -Siam1	AdaBoost	98.43	**86.64**
3.	Fuzz-Vec2im -Siam1	Random Forest	98.48	86.40
4.	Fuzz-Vec2im -Siam1	SVM	98.53	86.01
5.	Fuzz-Vec2im -Siam1	kNN - 1	**100**	84.29
6.	Fuzz-Vec2im -Siam1	Decision Trees	99.5	84.31
7.	Fuzz-Vec2im -Siam1	Linear Discriminant	98.69	85.38

Table 2 Confusion Matrixes of the best four performing ML models (LDA and AdaBoost) trained on the reduced feature space

		Class 1	Class 2	Per class accuracy
AB	Class 1	9323	388	96.00%
	Class 2	2624	10209	79.55%
			Overall accuracy	**86.64%**

		Class 1	Class 2	Per class accuracy
RF	Class 1	9336	375	96.14%
	Class 2	2692	10141	79.02
			Overall accuracy	**86.40%**

		Class 1	Class 2	Per class accuracy
NB	Class 1	9393	318	96.73%
	Class 2	2800	10033	78.18%
			Overall accuracy	**86.17%**

		Class 1	Class 2	Per class accuracy
SVM	Class 1	9400	311	96.80%
	Class 2	2842	9991	77.85%
			Overall accuracy	**86.01%**

b. Comparison with other feature selection / feature extraction techniques

A thorough comparative analysis between the proposed methodology and other well-known competing feature extraction / selection techniques is presented below.

- ***Experiments 8 – 9***: *Evaluating the effect of fuzzification on the proposed feature extraction methodology*

In experiments 8 – 9, we evaluated the effect of fuzzification on the performance of the proposed methodology. To accomplish this, we replaced the proposed fuzzification technique with a standard data normalization into the range [0, 1]. Both pre-processing techniques scale the data into the same range, however they have different characteristics:

- data normalization applies a linear transformation on the data rescaling all features to the same range, whereas

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

- the proposed fuzzification technique transforms raw data into class memberships (the generated fuzzy values declare at what extend a sample belongs to class 1).

The application of the fuzzy allocation scheme prior to feature extraction had a positive effect on both training and testing accuracies. Specifically, replacing the proposed fuzzification scheme with a standard data normalization technique lowered the testing accuracy by 6% (refer to experiment 9 in Table 3).

- ***Experiments 10 – 17***: *Comparing the proposed feature extraction approach with other FS techniques*

For comparison purposes, a wrapper FS technique was also employed to reduce the feature dimensionality of the initial 41-d space. This technique employs a search strategy to look through the space of possible feature subsets, evaluating each subset based on the quality of the performance of a given algorithm. A sequential forward selection strategy was implemented that starts with no feature and progressively adds one feature at a time. The same classifier was utilized in both the wrapper FS and the proposed vec2im+Siam methodology in order to set a fair comparison ground. The wrapper FS technique was implemented four times setting different termination criteria as follows:

- in the experiment 10, the wrapper FS was implemented to identify the most important feature from the entire feature space. The obtained 1-d space was compared with the 1-d space as it has been generated by the proposed in this paper methodology. A much lower testing accuracy was observed in experiment 10 (77.91%) indicating that the selected feature is less informative compared to the extracted feature from Vec2im-Siam.
- In experiments 12, 14 and 16, different termination criteria were set in the Wrapper FS technique leading to feature spaces of higher dimensionality (2-d, 3-d and 10-d, respectively). The discrimination capability of the resulted spaces was also compared with the 1-d space of the proposed methodology using the same classifier. The results verify that the extracted 1-d feature space of Vec2im-Siam is more descriptive than the resulted spaces of the Wrapper FS technique. This is even valid in the case of the 10-d feature space, in which the application of the same classifier led to an testing accuracy of 84.06 (exp.16) that is 2.6% lower than the testing performance of the proposed methodology implemented on a 1-d space.

To further evaluate the usefulness of the proposed fuzzification routine, we applied it as a pre-processing tool prior to the application of the Wrapper FS technique (refer to experiments 11, 13, 15 and 17) and finally compared the classification accuracy obtained with the one obtained in experiments 10, 12, 14 and 16 (where a standard data normalization is employed). Equal accuracies were obtained in experiments 10 and 11 (77.91 % in testing for both) and in experiments 12 and 13 (82.04% in testing for both). A slight increase was observed in the testing accuracy of experiments 15 and 17 (85.81% and 85.93%, respectively) compared to the accuracies achieved in experiments 14 and 16 (82.98% and 84.06%, respectively) indicating that the proposed fuzzification algorithm might also have a positive effect on a variety of other machine learning pipelines.

- ***Experiments 18 – 20***: *Comparing the proposed feature extraction approach with PCA*

The proposed Vec2im-Siam feature extraction methodology was finally compared with Principal Component Analysis (PCA) that is a well common feature dimensionality reduction approach. Three different feature spaces of varying dimensionality were generated via PCA as follows:

- A 1-d feature space using only the first extracted principal component (experiment 18)
- A 2-d feature space using the first two extracted principal components (experiment 19)
- A 3-d feature space using the first three extracted principal components (experiment 20).

The results highlighted the superiority of the Vec2im-Siam given that it outperformed PCA by more than 10% in testing even in the case in which higher dimensional spaces were used (e.g. in experiments 19 and 20).

Table 3 Comparative analysis with respect to competing feature

selection / extraction techniques using the same classifier (LDA)

Exp.	Preprocessing	Feature Extraction	Dimensionality of the resulted feature space	Training (%)	Testing (%)
8.	Fuzzification	Vec2im – Siam1	1	**98.69**	**86.64**
9.	Normalization	Vec2im – Siam1	1	90.20	80.64
10.	Normalization	FS (best feature)	1	82.84	77.91
11.	Fuzzification	FS (best feature)	1	82.84	77.91
12.	Normalization	FS (2 first features)	2	90.79	82.04
13.	Fuzzification	FS (2 first features)	2	90.79	82.04
14.	Normalization	FS (3 first features)	3	92.20	82.98
15	Fuzzification	FS (3 first features)	3	91.24	85.81
16.	Normalization	FS (10 first features)	10	92.33	84.06
17.	Fuzzification	FS (10 first features)	10	92.03	85.93
18.	Normalization	PCA (1 principal component)	1	89.69	75.64
19.	Normalization	PCA (2 principal components)	2	89.47	76.30
20.	Normalization	PCA (3 principal components)	3	89.48	76.32

4. Conclusions

A novel feature extraction pipeline is proposed in this paper that consists of the following components: (i) a fuzzy allocation scheme that transforms raw data to fuzzy class memberships, (ii) a mechanism for converting feature vectors to image (Vec2im) and (iii) a dimensionality reduction module that makes use of Siamese convolutional neural networks that finally reduces the input data dimensionality into a 1-d feature space. The proposed methodology was successfully applied on the NSL-KDD intrusion detection dataset. A thorough comparative analysis was performed that demonstrated the effectiveness of the different components of the methodology (e.g. the fuzzification, Vec2im including also visual interpretation) and also compared its performance with other well-known feature selection and extraction techniques. The proposed feature extraction methodology was assessed by applying a variety of ML models on the resulted 1-d feature space and evaluating their performances on the testing dataset. The increased discrimination capability of the resulted reduced feature space was verified by the fact that all competing models that were trained on it (either linear or not) had similar learning curves achieving similar testing accuracies within a small range of approximately 2%. The outcome of the proposed pipeline (Fuzz-Vec2im-Siam1) could be potentially used as a risk indicator for identifying cyber attacks, whereas the generated Vec2im images could be further utilized and/or integrated as a visual analytics tool in IDS. Future plans include the application of the proposed methodology in other sectors (such as in healthcare) where decisions should be taken based on complex and heterogenous data.

Authors' contributions

Serafeim Moustakidis designed the feature extraction pipeline, performed the experiments and drafted the manuscript. Patrik Karlsson participated in problem discussions and improvements of the manuscript.

Funding

This work has received funding from the European Community's H2020 Programme, under grant agreement Nr. 826183 (SPHINX).

Competing interests

The authors declare that they have no competing interests.

References

Alazab A, Hobbs M, Abawajy J, Alazab M (2012) Using feature selection for intrusion detection system. In Communications and Information Technologies (ISCIT), 2012 International Symposium on (pp.296-301). IEEE.

Atkeson C, Moore A, Schaal S (1997) Locally weighted learning. Artif Intell Rev 11:11–73.

Bajaj K, Arora A (2013) Dimension reduction in intrusion detection features using discriminative machine learning approach. IJCSI International Journal of Computer Science Issues 10(4):324–328

Belson W (1959) Matching and prediction on the principle of biological classification. Appl Stat. 8:65.

Bijone M (2016) A survey on secure network: Intrusion detection prevention approaches. Am J Inf Syst 20 (4):69–88.

Breiman L (2001) Random forests. Mach Learn 45:5–32.

Bromley J, Guyon I, Lecun Y, Sackinger E, & Shah R (1993) Signature verification using a Siamese time delay neural network. In J. Cowan, & G. Tesauro (Eds.), Advances in neural information processing systems (NIPS 1993) (Vol. 6). Morgan Kaufmann.

Butun I, Morgera S D, and Sankar R (2014) A survey of intrusion detection systems in wireless sensor networks. IEEE Communications Surveys Tutorials 16(1):266–282

Chebrolu S, Abraham A, and Thomas J P (2015) Feature deduction and ensemble design of intrusion detection systems Computers & Security 24(4):295–307, 6

Chopra S, Hadsell R, and LeCun Y (2005) Learning a similarity metric discriminatively, with application to face verification. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 539–546. IEEE.

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297.

DARPA dataset https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset. Accessed 15 May 2020

Dua S and Du X (2016) Data mining and machine learning in cybersecurity. CRC press

Duda R, Hart P, Stork D (2000) Pattern Classification, 2nd edn. Hoboken, NJ: Wiley Inter Science.

Elhag S, Fernández A, Bawakid A, Alshomrani S (2015) and Herrera F, On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. Expert Syst Appl 42(1):193–202

Freund Y, Schapire R (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55:119–139.

Hadsell R, Chopra S. & LeCun Y. (2006) Dimensionality Reduction by Learning an Invariant Mapping.. CVPR (2) (p./pp. 1735-1742), : IEEE Computer Society. ISBN: 0-7695-2597-0

KDD Cup 1999 http://kdd.ics.uci.edu/databases/kddcup99/ Accessed 15 May 2020

Khraisat A, Gondal I, Vamplew P (2018) An anomaly intrusion detection system using C5 decision tree classifier. In: Trends and applications in knowledge discovery and data mining. Springer International Publishing, Cham, pp 149–155

Kingma D P, Ba J (2014) Adam: A Method for Stochastic Optimization.. CoRR, abs/1412.6980.

Koch G R (2015) Siamese Neural Networks for One-Shot Image Recognition. Thesis

Kshetri N, Voas J (2017) Hacking power grids: a current problem. Computer 50(12):91–95

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

NSL-KDD data set for network-based intrusion detection systems https://www.unb.ca/cic/datasets/nsl.html. Accessed 15 May 2020

Potluri S, Ahmed S, Diedrich C (2018) Convolutional Neural Networks for Multi-class Intrusion Detection System. In Mining Intelligence and Knowledge Exploration; Springer: Cham, Switzerland, 2018; pp. 225–238.

Sahasrabuddhe A, Naikade S, Ramaswamy A, Sadliwala B, Futane P (2017) Survey on intrusion detection system using data mining techniques. Int Res J Eng Technol. 4(5):1780–4.

Scholkopf B (1997) Support Vector Learning. Munich: R. Oldenbourg Verlag.

Sharma S, Gupta R (2015) Intrusion detection system: A review. Int J Secur Its Appl 9:69–76.

Subramanian S, Srinivasan VB, Ramasa C (2012) Study on classification algorithms for network intrusion systems. Journal of Communication and Computer 9(11):1242–1246

Thaseen S and Kumar C A (2013) An analysis of supervised tree based classifiers for intrusion detection system. in 2013 international conference on pattern recognition, informatics and Mobile engineering, pp. 294–299

Witten I, Frank E, Hall M (2011) Data Mining. Burlington, MA: Morgan Kaufmann.

Xiao L, Wan X, Lu X, Zhang Y, and Wu D (2018) IoT security techniques based on machine learning, arXiv preprint arXiv:1801.06275

Zhang B, Yu Y, Li J (2018) Network Intrusion Detection Based on Stacked Sparse Autoencoder and BinaryTree Ensemble Method. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICCWorkshops), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.

Zhang H, Yu X, Ren P, Luo C (2019) Min, G. Deep Adversarial Learning in Intrusion Detection: A Data Augmentation Enhanced Framework. arXiv 2019, arXiv:1901.07949.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.