Unsupervised network traffic anomaly detection with deep autoencoders

VIBEKANANDA DUTTA[*], *Institute of Telecommunications and Computer Science, Bydgoszcz University of Science and Technology, al. Profesora Sylwestra Kaliskiego 7, 85-976 Bydgoszcz, Poland and Institute of Micromechanics and Photonics, Warsaw University of Technology, św. Andrzeja Boboli 8/507, 02-525 Warsaw, Poland.*

MAREK PAWLICKI[**], *Institute of Telecommunications and Computer Science, Bydgoszcz University of Science and Technology, al. Profesora Sylwestra Kaliskiego 7, 85-976 Bydgoszcz, Poland.*

RAFAŁ KOZIK[†], *Institute of Telecommunications and Computer Science, Bydgoszcz University of Science and Technology, al. Profesora Sylwestra Kaliskiego 7, 85-976 Bydgoszcz, Poland.*

MICHAŁ CHORAŚ[††], *Institute of Telecommunications and Computer Science, Bydgoszcz University of Science and Technology, al. Profesora Sylwestra Kaliskiego 7, 85-976 Bydgoszcz, Poland.*

Abstract

Contemporary Artificial Intelligence methods, especially their subset-deep learning, are finding their way to successful implementations in the detection and classification of intrusions at the network level. This paper presents an intrusion detection mechanism that leverages Deep AutoEncoder and several Deep Decoders for unsupervised classification. This work incorporates multiple network topology setups for comparative studies. The efficiency of the proposed topologies is validated on two established benchmark datasets: UNSW-NB15 and NetML-2020. The results of their analysis are discussed in terms of classification accuracy, detection rate, false-positive rate, negative predictive value, Matthews correlation coefficient and F1-score. Furthermore, comparing against the state-of-the-art methods used for network intrusion detection is also disclosed.

Keywords: Machine learning, deep learning, cybersecurity, intrusion detection system, autoencoder, deep neural network.

1 Introduction

1.1 Background

The conventional security tools have not been considered sufficient for the modern defense mechanisms due to the frequent changes in security definitions and lack of control over the security

[*]E-mail: vibekananda.dutta@pw.edu.pl
[**]E-mail: marek.pawlicki@utp.edu.pl
[†]E-mail: rkozik@utp.edu.pl
[††]E-mail: chorasm@utp.edu.pl

Vol. 30, No. 6, © The Author(s) 2022. Published by Oxford University Press.
Advance Access published 24 February 2022 https://doi.org/10.1093/jigpal/jzac002

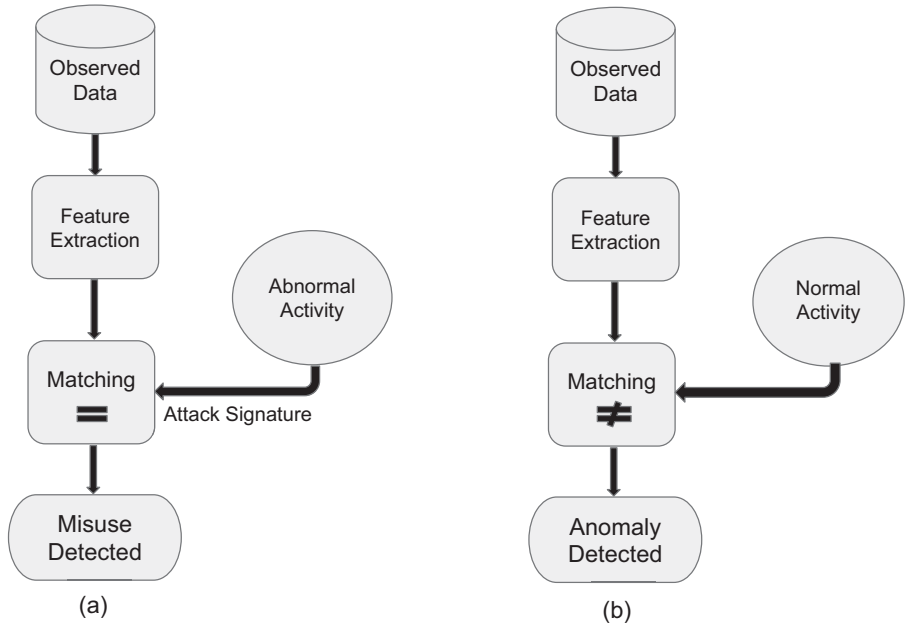

FIGURE 1. The figure depicts (a) signature-based threat detection and (b) anomaly detection approaches for malware detection. The figure is adopted from [8].

vulnerabilities. Network Intrusion Detection Systems (NIDSs) are considered well-known tools for monitoring and detection of malicious traffic in the Internet of Things (IoT) networks [14]. An Intrusion Detection mechanism captures network traffic in real-time and compares the received packet patterns to known patterns in order to detect the anomaly samples in the network. Moreover, the contemporary anomaly detection systems, even though used in practice, are often not stable because of the gradual change of definitions of what constitutes an anomaly [18].

The significant development of novel technologies in Information and Communication Technologies, like cloud computing, the IoT and Narrow-Band IoT (NB-IoT), comes with new vulnerabilities. The cost and high processing time of handling the traffic load is still a challenging tasks in NIDSs. These challenges increase the lack of trust concerning cybersecurity solutions, which many end-users already exhibit [38].

In this regard, many researchers have introduced several methodologies for defense mechanisms in order to detect intrusions in different environments, most notably in the application layer [5, 20], through NetFlows and other methods. The traditional method is often based on known malicious patterns. In this approach, the detection process matches the event pattern against the stored signatures (see Figure 1a). If a match is found, an intrusion signal is generated [17]. Currently, this constitutes the industry-standard approach to cyber attack detection referred to as the ‘signature-based’.

Anomaly detection methods are another group of protection mechanisms that acknowledge a malicious or unwanted behaviour that is compared against a set of normal traffic samples. Behaviour can be governed by observing network activities, processes and resource changes. The advantage of anomaly detection methods is their ability to detect attacks that did not take place in the past (the so-called zero-day exploits), which groups them in the predictive methods (see Figure 1b). To do so,

firstly, the pattern of normal traffic must be established and then matched versus the current traffic samples [21]. Whenever there is no match, an alarm is raised.

An increasing collection of works utilizing machine learning techniques to deal with malware detection can be found in the literature [4]. The set of researched methods is varied, covering both supervised and unsupervised learning. Methods like Bayesian networks [2], clustering algorithms (K-Means, Fuzzy C-means, etc.) [32], self-organizing maps [19], and one-class support vector machines (SVM) [34] calculate the distribution of normal network data and define any data that diverge from the normal distribution as an anomaly. On the other hand, SVMs, decision trees [13] and k-nearest neighbour (KNN) [7] and similar methods use labelled data to build classifiers to detect intrusions in a supervised fashion.

Deep learning methods, such as AutoEncoders (AE), Deep Neural Networks (DNN), Deep Belief Networks (DBNs), Recurrent Neural Networks (RNN) and Convolutional Neural Networks, are gaining popularity in automatic feature extraction and classification [37].

Some deep learning approaches can automatically extract high-level features without manual formulation [36]. The last category uses various ensemble and hybrid techniques to improve the detection performance. These include different classification method combinations [1]. However, further improvements are still necessary due to a high false-positive rate [37]. This paper is an extended version of the work presented in [8].

1.2 Motivation and research challenges

The heterogeneous nature of IoT devices is one of the sources of security vulnerabilities. Many IoT devices are have doubtful security standards, such as unauthenticated telnet ports, obsolescent firmware and unencrypted transmission of sensitive data, etc [30].

The objective of the deep learning-based NIDS tools for IoT networks is to facilitate attack detection capabilities to protect a network. Many intrusion detection tools proposed and evaluated in the literature are tested on outdated datasets, which do not reflect real-world IoT network traffic [10]. Fairly often, the optimization of the algorithm hyperparameters is completely omitted.

Adopting conventional security information in a large organization is challenging due to the growing volume of collected data and an increasing number of heterogeneous sources producing logs/alarms at various data rates. A diverse established benchmark dataset containing information that reflects all the strategies of real-world attacks is essential for the successful deployment of machine learning methods for IoT security.

NIDS relies on machine learning algorithms that fit data. The data was collected in one network, but the IDS will have to be deployed in a different network with similar accuracy.

1.3 Research contributions and article roadmap

In NIDS, the subjects of volume, velocity and variety of data in contemporary network traffic (also known as the V's of Big Data) are challenges that require significant attention. This work evaluates a proposed hybrid deep learning mechanism to detect network intrusion, using the formulated datasets: UNSW-NB15 and NetML-2020.

Therefore, this research attempts to evaluate a number of decoder topology setups to detect network anomalies in an unsupervised fashion. This work will be further continued within the H2020 InfraStress project.

This article offers the following contributions:

- Following [10], this work incorporates flow-level analysis and class balancing that combines Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbours (ENN) approach to enhance the classification efficiency;
- The research is an extension of [8], leveraging different arrangements of neural networks trained in an unsupervised fashion to detect network anomalies;
- It applies a dimensionality reduction approach - a Deep AutoEncoder (DAE) for reducing the size of the input feature vector to the classifier in order to reduce the computational complexity;
- The disclosed set of techniques is put through a series of experiments to illustrate the extent of improvements in the detection rate using the chosen datasets in comparison to several other state-of-the-art methods.

The rest of the paper discusses these contributions in-depth. Section 2 presents the related works, in particular the recent unsupervised approaches in network intrusion detection systems. In Section 3, a summary of the proposed architecture and its main contributions are outlined. Sections 4 and 5 delineate the details of the setting of the study and the obtained results. Finally, Section 6 ends with conclusions, and the possible future research.

2 Related work

The continuous advancements in networking technologies have emphasized the significance of a defense mechanism capable of identifying zero-day attacks [11]. With this background in mind, the concepts of intrusion detection systems (IDS) and anomaly detection mechanisms are promising defensive tools in the cybersecurity domain [35]. This topic has engaged a substantial amount of research attention recently in the field of computer network security. However, an anomaly detection system may categorize previously unseen behaviour as anomalies [15], which results in high false-positive rates. Thus, an effective IDS that handles massive network data with changing patterns in real-time scenarios is desirable [12, 22].

The technical advances in the application of neural networks for intrusion detection have been a promising area of research recently. Dutta *et al.* introduced an anomaly detection tool: (i) a deep sparse AE, (ii) a DNN and (iii) a LSTM followed by a logistic regression classifier. These were employed to detect anomalous traffic in IoT environments. In [33], Tang *et al.* developed an intrusion detection defense tool employing DNNs for flow-based anomaly traffic analysis in software-defined networking environments. A different approach is investigated by Zhang *et al.* in , where multi-dimensional feature fusion and an ensemble learning approach is handled by different base classifiers. Most recently, Kwon *et al.* scrutinized several deep learning-based defense mechanisms for anomaly detection, including the Restricted Boltzmann Machine, DBNs, DNNs, RNNs and AEs.

The majority of high-accuracy results achieved in IDS are proposed as supervised tasks, which prerequisite adequate, labelled network data in the training phase [35]. In real-world IoT network environment, labelling network traffic data is laborious and error-prone. Bearing these limitations in mind, the development of an effective NIDS, taking into account unsupervised deep learning methods is desirable from the practical standpoint.

Among unsupervised methods, AE have been explored for robust feature representation from a large volume of data and exhibit strong capability for data reconstruction [33]. For the AE-variant approach, a deep Convolutional AE is introduced in to exhibit shift-invariant sparse representations for anomaly detection. For instance, in [29], Hassan *et al.* developed the sparse AE, which optimizes the hyperparameters to facilitate a better ability to extract useful features and classify malicious

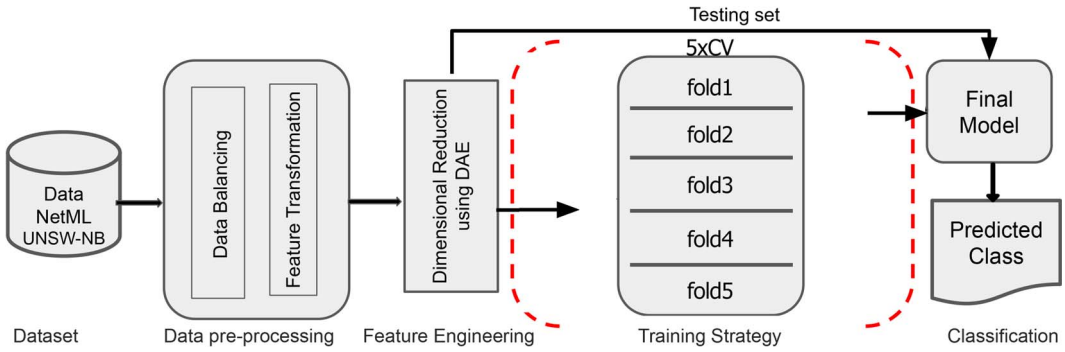

FIGURE 2. The figure depicts the implementation strategies for malware detection.

attacks. Furthermore, Ieracitano *et al.* [16] explored a statistical analysis and AE-driven intelligent IDS. The proposed method combines data analytics and statistical techniques with contemporary machine learning algorithms to extract more optimized, strongly correlated features.

The authors of [26] employed conditional variational AEs to detect and classify the five types of attack labels: (i) normal, (ii) DoS, (iii) R2L, (iv) Probe and (v) U2R. Moreover, given a labelled training NSL-KDD dataset, the decoding network of the proposed model is additionally trained with the class labels associated with the trained data logs. Meidan *et al.* [27] explored DAEs to detect botnet attacks in the IoT environment. The authors employ a set of AEs, each AE learns the normal network behaviour of the designated IoT device and detects anomalous traffic from the device.

The literature review reveals that several AE-variant models and their ensembles are employed to detect network anomalies in traffic in an unsupervised fashion. The majority of research pieces report promising results in various evaluation metrics.

3 Proposed method

The objective of the proposed network intrusion detection method is to obtain a reliable classification of malicious patterns by leveraging different arrangements of DAEs. The conceptual overview of the proposed strategy is depicted in Figure 2. The framework incorporates the following: (i) the data that constitutes contemporary network traffic and real-world attacks; (ii) the implementation of data preprocessing and feature transformation. This step also balances the distribution of classes in the dataset; (iii) dimensionality reduction with the use of DAE; (iv) a deep decoder network that uses the latent representation of features to reconstruct the input data; (v) multiple topologies of deep decoders are evaluated; (vi) the outcome of anomaly detection. Those steps are thoroughly elaborated upon in the forthcoming paragraphs.

3.1 Pre-processing

The disproportionate ratio of samples in each class distribution is a common obstacle in machine learning classification. Referred to as the imbalance problem, this scenario affects performance of many ML algorithms. The model trained with imbalanced data often classifies the minority class samples as the majority class samples [16].

Therefore, the count of samples in each class (or label) in the dataset is subjected to the balancing procedure. Following the findings of our earlier research [10, 24] and a run of preliminary experiments, to deal with the imbalance problem, a method that leverages SMOTE and ENN was used. It levels class distributions by increasing the minority class instances with the use of an adaptation of the KNN algorithm. First, SMOTE is applied to create synthetic data points of minority class samples, then using ENN, the data points on the border or boundary are removed to increase the separation of the two classes.

After applying the data balancing approach to enhance the processed data, the artificially balanced samples were subjected to the standardization process. The samples were standardized by removing the mean and scaling to unit variance by employing popular python library- **scikit-learn StandardScaler**. Furthermore, the input values to our DAE must be a real vector, so each symbolic feature is finally converted to a numerical feature. In order to achieve this task, we applied One-Hot-Encoding and Label-Encoding.

3.2 Dimensionality reduction

Since the performance of a classifier highly depends on the selected features, the problem consists in finding the most relevant features to maximize its performance. Principal Component Analysis (PCA) and AE are widely used feature engineering approaches for dimensionality reduction. However, PCA is restricted to a linear map and AEs are capable of modelling complex non-linear functions.

The topic of discussion in this section concerns the construction of a new n-dimensional feature space that explains the data of initial dimensional feature space.

Following the research conducted in , we have utilized a DAE to decrease the number of features. The AE can formulate a latent representation of the inputs. The used DAE is organized into a pair of two connected sub-networks: an encoder (e_θ) and a decoder (d_θ), in addition to the k-layer between the sub-networks, called the bottleneck, or the code layer. The DAE maps the input data $x_i \in \Re^n$ to its code layer representation $\hat{x}_i \in \Re^r$ through its encoder sub-network. The training algorithm updates the weights $\{W_e, W_d\}$ and biases $\{b_e, b_d\}$ to reduce the value of error from cost function at the outputs of the decoder sub-network.

$$h_i = e_\theta(x_i). \tag{1}$$

The decoder sub-network reconstructs the encoding back to its the original input state,

$$\hat{x}_i = d_\theta(h_i). \tag{2}$$

The parameters for the sub-networks (e_θ, d_θ) are calculated simultaneously by minimizing the loss function.

This stage is essential for optimizing the average reconstruction error (RE). In order to minimize the RE, this work utilized the mean square error. The initial DAE training stage will constitute a good starting point the training of multiple decoder topologies, which will facilitate anomaly detection.

3.3 Classifier modeling

A range of neural networks are constructed employing multiple computational layers (i.e. hidden layers). The DNNs will perform the function of decoders—train on an autoencoded representation of the input vector and attempt reconstruction of the input vector.

$$\hat{y} = \sigma(W.\hat{x} + b) \tag{3}$$

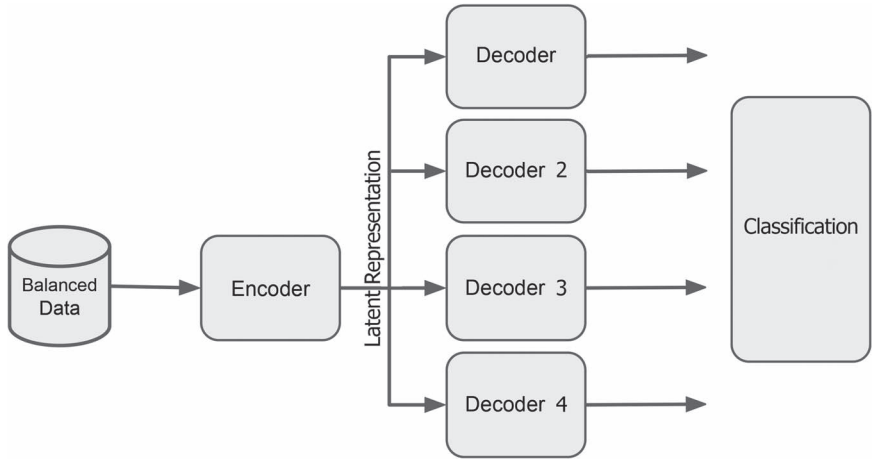

FIGURE 3. The training stage of the decoders.

σ refers to the neural activation function. Key to our technique is the learning strategy that exploits the direct utilization of the weight of the sub-network (i.e. encoder) trained in the feature engineering stage as a starting point to the classification phase.

Training the encoder separately allows for reduced complexity while training decoders and ensures uniform latent representation as the input to the classification stage. The objective of the loss function l_{cf} delineates the difference between the ground truth test samples (y) and the predicted samples ($\hat{y}$).

The training of decoder topologies (see Figure 3) is unsupervised. The decoders are trained separately by taking the latent representation as the input.

In this work, the experimental evaluation of the proposed topologies is performed. The new propositions are stacked against methods previously established in , using the UNSW-NB15 dataset. The results are disclosed in Table 6

This work showcases the performance of different decoder network topology setups, trained using the latent representation of features coming from the bottleneck part of a DAE trained in a preceding phase. Each setup can have its own influence on the classification capabilities and training time. The detailed description, which covers the multiple network topology configurations and a set of applied hyperparameters, is disclosed in next section.

3.4 Configuration and tuning of hyperparameters

The following paragraphs elaborate on the multitude of configurations possible for different network setups. Our role in this work was to explore the perspective of learning scheme on multiple network topology setups.

Each topology setup from the given Table 1 has an influence on the classification capabilities and computational complexities. Therefore, this work scrutinized multiple (in our work it is 4 topology setups) to verify and demonstrate the effective scheme on two established benchmark datasets. For completeness of the configuration, we additionally explore three neural activation functions: (i) Rectified Linear Unit (ReLU), (ii) Hyperbolic Tangent (tanh), (iii) Sigmoid and their optimal

TABLE 1. Hyperparameters for optimal model performance in different settings.

Hyperparameters	Experimented values	Selected configuration			
		$Model_1$	$Model_2$	$Model_3$	$Model_4$
Number of layers	2–5	3	3	5	3
Neurons per layer	20–70	s_1	s_2	s_3	s_4
Epochs	500, 700, 1k	500	700	500	1k
Batch size	64, 128, 256	128	128	64	256
Optimizer	Adam (a), rmsprop (r)	r	r	a	r
Dropout rate (lr)	0.01, 0.1, 0.2, 0.3	0.3	0.01	0.3	0.3

*Selected configuration for each model.

network setup over the hyperparameter's space. The parameters included (i) iteration count, (ii) batch size, (iii) optimizer and (iv) loss function, respectively.

The hyperparameter configuration of the first setup (s_1) incorporates the ReLU activation function, a hidden layer with the ReLU activation function followed by a dropout layer with the dropout set to 0.2. Finally, the topology setup closes with an output layer using the Sigmoid activation function. The loss function was set to 'binary_crossentropy'. The optimizer was set to Root Mean Square Propagation (rmsprop).

The next ANN setup (s_2) follows a similar pattern to the fist setup, adding two dropout layers, both set to 0.1. The first dropout layer was set after the input layer, and the next one was set right after the hidden layer.

The third setup (s_3) is unlike the first two networks: an input layer, three hidden layers, three dropout layers and an output layer. The first hidden layer incorporates a number of neurons set to 46 and the tanh activation. Next, both the second and third hidden layers with a set of 35 and 27 neurons and the tanh activation function. The dropout layers with the dropout set to 0.01 were used after each hidden layer. The chosen optimizer was Adaptive Momentum Estimation (Adam).

In the final network setup (s_4), there is a dropout layer with dropout set to 0.3 right after the input layer. This is followed by a hidden layer of 27 neurons and again a dropout layer. Finally, the setup closes with 'Sigmoid', and the Adam optimizer was applied.

The forecast of the classification is fixed to binary and is expressed by the sign of the value that is the result of the classifier. For the update of all the weights and biases, rmsprop and Adam optimizers were evaluated. The sets of s_1, s_2, s_3 and s_4 are different numbers of neurons per layer. A summary of each topology setup configuration can be found in Table 1, while a short description of the purpose and the setup of each configuration is provided below.

Using the findings presented in [27], the authors employed the grid search hyperparameter tuning method to find the optimal set of hyperparameters. The grid search is exhaustive search method. The search space included the set of neurons, the used epochs count, the batch size, the optimizer, and the activation function to achieve the highest accuracy. Finally, the most successful set of hyperparameters that offer the best classification accuracy achieved by each network setup is presented in Table 1.

To ensure that the learning process is meaningful, we evaluate each topology setup via k-fold (k =5) cross-validation in order to obtain the best overall performance. This is shown in Tables 2 and 3.

The experiments shown in Tables 2 and 3 have demonstrated that the training of each topology setup generally yields similar performance irrespective of the individual dataset fold. Using a

TABLE 2. Results of evaluated classifier performance reported for each fold of the 5-fold CV on UNSW-NB15 dataset.

Fold	$Model_1$	$Model_2$	$Model_3$	$Model_4$
F1	0.9612	0.9166	0.9976	0.9987
F2	0.9970	0.9876	0.9990	0.9989
F3	0.9249	0.9494	0.9985	0.9991
F4	0.9127	0.8769	0.9994	0.9088
F5	0.9996	0.9963	0.9992	0.9995
Mean acc.	0.9590	0.9453	0.9987	0.9810
Standard dev.	3.578	4.451	0.064	3.610

TABLE 3. Results of evaluated classifier performance reported for each fold of the 5-fold CV on NetML-2020 dataset.

Fold	$Model_1$	$Model_2$	$Model_3$	$Model_4$
F1	0.9768	0.9791	0.9842	0.9961
F2	0.9929	0.9949	0.9854	0.9953
F3	0.9942	0.9922	0.9959	0.9942
F4	0.9925	0.9955	0.9910	0.9954
F5	0.9945	0.9966	0.9815	0.9950
Mean acc.	0.9901	0.9916	0.9876	0.9952
Standard dev.	0.0673	0.0644	0.517	0.0616

separate training stage for the encoder leads to a significant decrease in the number of parameters and thus speeding up the training.

In this sense, our per-modality pre-training step, when the network itself is divided into meaningful parts, that are adjusted separately and then combined, can be considered as the first step of this model growing process. The results of the undertaken experiments presented the accuracy for each particular setup.

4 Experimental setup and data

This work utilizes the TensorFlow open-source library, developed by the Google Brain team. The library offers machine and deep learning tools and is widely applied by various scientific applications. Along with TensorFlow, the work uses scikit-learn stack, and a set of other machine learning libraries. The experiments were performed on a Linux setup with 16 GB RAM and the Intel Core i710-th Generation Processor.

4.1 Dataset description

In this section, the used datasets created specifically for the evaluation of network intrusion/anomaly detection using various machine learning methods are described.

The UNSW-NB15 [25] is a newly established time-based dataset that contains synthetic traffic with contemporary attacks. The TCP-dump tool was used to capture 100 GB of raw traffic.

TABLE 4. Comparison of different performances using the UNSW-NB15 test subset.

Model	Acc.	Pr	Re	FPR	F-score	NPV
$Model_1$	0.9985	0.9798	0.9700	0.19	0.9749	0.8490
$Model_2$	0.9707	0.9694	0.9694	0.41	0.9596	0.8350
$Model_3$	0.9822	0.9940	0.9990	0.1750	0.9965	0.9091
$Model_4$	0.9770	0.9899	0.9980	0.2000	0.9939	0.8733

TABLE 5. Comparison of different performances using the NetMl-2020 test subset.

Model	Acc.	Pr	Re	FPR	F-score	NPV
$Model_1$	0.9994	0.9990	0.9980	0.0909	0.9985	0.9333
$Model_2$	0.9996	0.9880	0.9787	0.0809	0.9880	0.9091
$Model_3$	0.9996	0.9983	0.9986	0.0170	0.9988	0.9400
$Model_4$	0.9996	0.9999	0.9990	0.0009	0.9994	0.9574

Twelve algorithms and tools such as Argus, Bro-IDS were employed to generate UNSW-NB15. The dataset is divided into a training set and a test set according to the hierarchical sampling method, namely, UNSW-NB15-training-set.csv and UNSW-NB15-testing-set.csv. UNSW-NB15 features are categorized into five distinct categories: (i) flow features, (ii) basic features, (iii) content features, (iv) time features and (v) additional generated features. The dataset has nine attack types and 49 features including a class label. The attacks in this dataset include fuzzer, analysis, backdoor, DoS, exploit, generic, reconnaissance, shellcode and worm attacks. The attributes are mixed in nature, with some being nominal, some being numeric and some taking on time-stamp values.

To validate the performance of the applied framework, a newly established benchmark dataset NetML-2020 is employed. The set was created for anomaly detection tasks in the early 2020. The Netflow features are extracted by providing a raw pcap file as an input to the feature extraction tool. Each Netflow sample is listed in the output file in JSON format. Lastly, a unique id number is defined to disclose every flow feature and the label information by referring to the raw traffic packet capture file. The NetML dataset incorporates 484,056 Netflows and 48 feature attributes. Three annotations, 'top-level', 'mid-level' and 'fine-grained', are chosen to address the class granularity. This work considers the 'top-level' granularity, therefore, only 26 meta-features were chosen after the feature transformation stage.

5 Results

This section discusses and analyses the empirical performance of the proposed strategies. The work commenced with preliminary experiments using both UNSW-NB15 and NetML datasets.

In the experiments, the precision, recall, F-measure, false positive rate, negative balance error rate, negative predictive value and accuracy are measured, which are calculated from the confusion matrix. We also provide the values of negative predictive value (NPV), respectively.

The results of this phase are summarized in Tables 4 and 5.

This work performed comparative studies using the existing state-of-the-art methods on comparable test subsets. Tables 6 and 7 illustrate the effect of the results attained by the top two topology setups against individual classifiers, namely DNN [8], random forest [8], SVM [10] and Ensemble [10]. The UNSW-NB15 and NetML-2020 datasets were utilized to assess the efficiency of the

TABLE 6. Comparison of different state-of-art methods and the proposed two best topology setups on the UNSW-NB15 dataset.

Model	Acc.	FPR	NPV	MCC	Data balance
RF	0.8514	0.1396	0.8139	–	No
DNN	0.8815	0.1069	0.8357	–	No
Hybrid	0.9129	0.0859	0.8889	–	No
Ensemble	0.9993	0.0146	0.9804	0.9794	Yes
$Model_1$	0.9985	0.0550	0.9091	0.9410	Yes
$Model_3$	0.9822	0.070	0.8733	0.9289	Yes

TABLE 7. Comparison of different state-of-art methods and the proposed two best topology setups on the NetML-2020 dataset.

Model	Acc.	FPR	NPV	MCC	Data balance
RF	0.961	0.33	0.89	0.920	No
SVM	0.961	0.42	0.8712	0.927	No
DNN	0.988	0.20	0.9299	0.97	No
Ensemble	1.00	0.0	0.9979	1.0	Yes
$Model_3$	0.9996	0.0109	0.9491	0.9876	Yes
$Model_4$	0.9996	0.009	0.9574	0.9913	Yes

classifiers. Furthermore, the evaluation results taking into account the accuracy (Acc.), FPR, NPV and Matthews correlation coefficient (MCC) [31] are presented in Tables 6 and 7.

This illustration reflects the fact that the established ensemble method presented in [10] achieved so far the highest accuracy by reaching 0.9993 and 1.00, as well as minimal false detection, e.g. 0.00 and 0.0146 against the four distinct topology setups. To further summarize, the achieved results and analysing the statistical significance of the outcomes by individual setups demonstrated that neither inflating the number of neurons in layers, nor exploiting the network topology setup contribute to increasing the classification accuracy for the evaluated benchmark datasets.

6 Conclusions and future work

In this work, a defense mechanism for the task of network intrusion detection that leverages unsupervised learning with DAEs is evaluated. From a modeling perspective, the work presented in this article has demonstrated that conceptual architectures of four different network topology setups and the training strategy that are particularly efficient in learning and exploring the potential sparse representations of network data features. The employed strategy saves resources, especially in a big data scenario containing modern attacks.

The trained DAE is employed to formulate the latent representation of features, which is used to train the deep decoders. The problem of data balancing is also tackled followed by an appropriate data preprocessing stage. The proposed method provided good performance in distinguishing between attack and normal activities. The results of the chosen network setups are disclosed. Thorough analyses were performed with the use of two established benchmark datasets, namely UNSW-NB15 and NetML-2020. The experiments have proven, that significant accuracy can be achieved using unsupervised methods for network anomaly detection (see Tables 6 and 7).

Considering future work, we plan to address the problem of classifying multiple attack families. We also wish to study an effective way to improve the detection performance of minority attacks.

Acknowledgements

This work is funded under InfraStress project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 833088.

Data Availability Statement

The datasets used in this work are publicly available for research purposes.

- UNSW-NB15: shorturl.at/deDKO
- NetML-2020: https://github.com/ACANETS

References

[1] A. A. Aburomman and M. B. I. Reaz. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers & Security*, **65**, 135–152, 2017.

[2] S. Aljawarneh, M. Aldwairi and M. B. Yassein. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, **25**, 152–160, 2018.

[3] A. Binbusayyis and T. Vaiyapuri. Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class svm. *Applied Intelligence*, 1–15, 2021.

[4] L. Caviglione. *et al.* Tight arms race: overview of current malware threats and trends in their detection. in *IEEE Access*, **9**, pp. 5371–5396, 2021, https://doi.org/10.1109/ACCESS.2020.3048319.

[5] M. Choraś and R. Kozik. Machine learning techniques applied to detect cyber attacks on web applications. *Logic Journal of the IGPL*, **23**, 45–56, 2015.

[6] M. Choraś and M. Pawlicki. Intrusion detection approach based on optimised artificial neural network. *Neurocomputing*, **452**, 705–715, 2021.

[7] Y. Djenouri, A. Belhadi, J. C.-W. Lin and A. Cano. Adapted k-nearest neighbors for detecting anomalies on spatio–temporal traffic flow. *IEEE Access*, **7**, 10015–10027, 2019.

[8] V. Dutta, M. Choraś, R. Kozik and M. Pawlicki. Hybrid model for improving the classification effectiveness of network intrusion detection. In *The 13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pp. 405–414. Springer, 2020.

[9] V. Dutta, M. Choraś, M. Pawlicki and R. Kozik. Detection of cyber attacks traces in IoT data. *Journal of Universal Computer Science*, **26**, 1422–1434, 2020.

[10] V. Dutta, M. Choraś, M. Pawlicki and R. Kozik. A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors*, **20**, 4583, 2020.

[11] V. Dutta and T. Zielinska. Networking technologies for robotic applications. *International Journal of Advanced Studies in Computer Science and Engineering*. **4**, 45–51, 2015. https://doi.org/10.1109/ICCS.2015.35.

[12] F. Farahnakian and J. Heikkonen. A deep auto-encoder based approach for intrusion detection system. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pp. 178–183. IEEE, 2018.

[13] R. Ganeshan and S. Paul Rodrigues. I-AHSDT: intrusion detection using adaptive dynamic directive operative fractional lion clustering and hyperbolic secant-based decision tree classifier. *Journal of Experimental & Theoretical Artificial Intelligence*, **30**, 887–910, 2018.
[14] K. Hashizume, D. G. Rosado, E. Fernández-Medina and E. B. Fernandez. An analysis of security issues for cloud computing. *Journal of Internet Services and Applications*, **4**, 5, 2013.
[15] F. Hosseinpour, P. V. Amoli, F. Farahnakian, J. Plosila and T. Hämäläinen. Artificial immune system based intrusion detection: innate immunity using an unsupervised learning approach. *International Journal of Digital Content Technology and its Applications*, **8**, 1, 2014.
[16] C. Ieracitano, A. Adeel, F. C. Morabito and A. Hussain. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing*, **387**, 51–62, 2020.
[17] A. Jain, B. Verma and J. Rana. Anomaly intrusion detection techniques: a brief review. *International Journal of Scientific & Engineering Research*, **5**, 1372–1383, 2014.
[18] G. R. Jidiga and P. Sammulal. Anomaly detection using machine learning with a case study. In *The 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 1060–1065. IEEE, 2014.
[19] A. Karami. An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities. *Expert Systems with Applications*, **108**, 36–60, 2018.
[20] R. Kozik and M. Choraś. Protecting the application layer in the public domain with machine learning methods. *Logic Journal of the IGPL*, **27**, 149–159, 2019.
[21] R. Kozik, M. Choraś, A. Flizikowski, M. Theocharidou, V. Rosato and E. Rome. Advanced services for critical infrastructures protection. *Journal of Ambient Intelligence and Humanized Computing*, **6**, 783–795, 2015.
[22] R. Kozik, M. Pawlicki and M. Choraś. A new method of hybrid time window embedding with transformer-based traffic data classification in iot-networked environment. *Pattern Analysis and Applications*, 1–9, 2021.
[23] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim and K. J. Kim. A survey of deep learning-based network anomaly detection. *Cluster Computing*, **22**, 949–961, 2019.
[24] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT. *Sensors*, **17**, 1967, 2017.
[25] A. Mahfouz, A. Abuhussein, D. Venugopal and S. Shiva. Ensemble classifiers for network intrusion detection using a novel network attack dataset. *Future Internet*, **12**, 180, 2020.
[26] A. Makhzani and B. Frey. Winner-take-all autoencoders. arXiv preprint, arXiv:1409.2752, 2014.
[27] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher and Y. Elovici. N-baIoT–network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, **17**, 12–22, 2018.
[28] N. Moustafa and J. Slay. The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, **25**, 18–31, 2016.
[29] H. Musafer, A. Abuzneid, M. Faezipour and A. Mahmood. An enhanced design of sparse autoencoder for latent features extraction based on trigonometric simplexes for network intrusion detection systems. *Electronics*, **9**, 259, 2020.
[30] D. Patel, K. Srinivasan, C.-Y. Chang, T. Gupta and A. Kataria. Network anomaly detection inside consumer networks–a hybrid approach. *Electronics*, **9**, 923, 2020.

[31] M. Pawlicki, M. Choraś, R. Kozik and W. Hołubowicz. On the impact of network data balancing in cybersecurity applications. In *Computational Science – ICCS 2020: 20th International Conference, Amsterdam*, The Netherlands, June 3–5, 2020, Proceedings, Part IV, pp. 196–210. Springer, 2020.

[32] W. Shang, J. Cui, C. Song, J. Zhao and P. Zeng. Research on industrial control anomaly detection based on FCM and SVM. In *The 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 218–222. IEEE, 2018.

[33] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi and M. Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *The 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 258–263. IEEE, 2016.

[34] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang and Q. Qu. Ramp loss one-class support vector machine; a robust and effective approach to anomaly detection problems. *Neurocomputing*, **310**, 223–235, 2018.

[35] T. Vaiyapuri and A. Binbusayyis. Application of deep autoencoder as an one-class classifier for unsupervised network intrusion detection: a comparative evaluation. *PeerJ Computer Science*, **6**, e327, 2020.

[36] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viégas and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensor flow. *IEEE Transactions on Visualization and Computer Graphics*, **24**, 1–12, 2017.

[37] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou and C. Wang. Machine learning and deep learning methods for cybersecurity. *IEEE Access*, **6**, 35365–35381, 2018.

[38] Y. Yang, K. Zheng, C. Wu and Y. Yang. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors*, **19**, 2528, 2019.

[39] H. Zhang, J.-L. Li, X.-M. Liu and C. Dong. Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. *Future Generation Computer Systems*, **122**, 130–143, 2021.

Received 20 February 2021