Identifying Novelty in Network Traffic

Joshua Sylvester
School of Computing
University of Kent, UK
Email: jrs71@kent.ac.uk

Rogério de Lemos
School of Computing
University of Kent, UK
Email: r.delemos@kent.ac.uk

***Abstract*—In a typical Security Operations Centre (SOC), detection methods for malicious transactions are usually resource-intensive, requiring a large team to monitor traffic, which is not ideal for efficient and effective decisions. This paper presents the MAE-NAE FRAMEWORK, consisting of two autoencoders and an adjudicator, which is fast and accurate, but not resource-intensive. One autoencoder is trained on malicious data, while the other is trained on normal data. The adjudicator classifies transactions into malicious, normal or novel, depending on the confidence level. Although autoencoders are widely used for novelty detection, they have not been used to identify novelty in network traffic, which is the key goal of MAE-NAE FRAMEWORK. This allows the provision of a triage system that identifies transactions as novel for which the confidence level in classifying either normal or malicious is low. For evaluating the MAE-NAE FRAMEWORK, we have used the KDDCUP99 benchmark dataset with a simple linear adjudicator. The MAE-NAE FRAMEWORK can classify 94.73% of data as normal or malicious leaving 5.27% of the transactions as novel. We have compared our solution against various solutions within the literature, and the MAE-NAE FRAMEWORK is more effective in classifying transactions.**

I. INTRODUCTION

Although several approaches exist for detecting malicious transactions, a key challenge in protecting systems against attacks is novelty identification in network traffic. This is particularly critical in Security Operations Centres (SOC), which may receive millions of transactions per second. Hence, there is a need for efficient and effective solutions capable of identifying both normal and novel malicious transactions. Existing solutions for detecting malicious transactions rely on machine learning techniques, however, most of these solutions cannot detect transactions that may be novel. These solutions tend to work for a short period but decay over time because of concept drift - a reality when dealing with network attacks. Moreover, in binary classifiers, novel transactions may be captured as edge conditions, leading to misclassifications, thus losing what might be a potential novel transaction. By detecting novelty, concept drift can potentially be identified through an increase in novel transactions.

This paper presents a study on using autoencoders to identify novelty in network traffic. Autoencoders have been widely used to detect novelty based on reconstruction errors and shown to outperform other techniques [1]. The dimensionality of the data is reduced, and then the input is reconstructed from this compressed form. The reconstruction error is high for classes in which the autoencoder is not trained. To identify novel network transactions and detect both normal and malicious ones, a Malicious Autoencoder-Normal Autoencoder Framework (MAE-NAE FRAMEWORK) is proposed. It uses two autoencoders and an adjudicator to classify the network traffic into three classes: normal, malicious and novel. One autoencoder is trained on normal network traffic, while the other is on malicious network traffic. The reconstruction error is then passed to the adjudicator which classifies the transaction based on these errors. The MAE-NAE FRAMEWORK has demonstrated to be an effective solution by detecting novel network transactions in an ever-changing environment. It achieved an F1 Score of 0.981 and was able to classify 94.73% of the data achieving better accuracy than other state-of-the-art techniques.

The rest of the paper is organised as follows. In the next section, the principles associated with MAE-NAE FRAMEWORK are introduced. Section III describes the methodology for realising the proposed framework. Section V details the experiments performed and the results obtained. Section VI evaluates the framework by comparing it with other similar approaches. Then, Section VII presents some related work. Finally, we conclude the paper and indicate future lines of research.

II. MAE-NAE FRAMEWORK

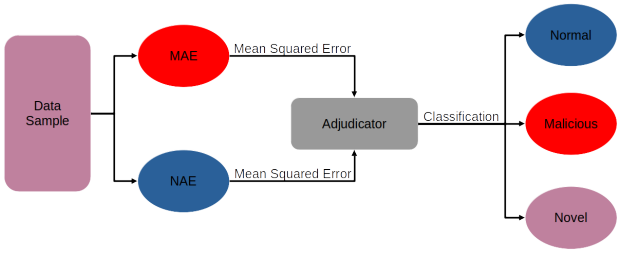

Fig. 1. Structure of MAE-NAE FRAMEWORK

The overall MAE-NAE FRAMEWORK can classify transaction data into normal, malicious and novel classes. Novelty detection is important as it allows the use of other techniques for classifying transactions that could not be classified as malicious or normal with a degree of confidence. The notion of novelty in system transactions is important since data, over time, can be affected by concept drift.

Fig.1 shows the overall structure of the MAE-NAE FRAMEWORK, with two autoencoders: an MAE and an NAE

trained on malicious and normal transactions, respectively. The adjudicator receives the outputs from the two autoencoders, as Mean Squared Error (MSE), which are used for classifying transactions. The motivation to use this structure is to maintain a dynamic solution that is more loosely coupled and can therefore be adapted separately for improving the detection of novelty. For example, if the performance of the NAE is degrading, it can be switched out in a modular fashion instead of implementing a whole new system.

In the following sections, the structure of the MAE-NAE FRAMEWORK, in terms of its autoencoders and adjudicator, is described in more detail.

A. Autoencoders

An autoencoder, originally proposed by Baldi [5], is an artificial neural network that learns from efficient data encodings. The autoencoder consists of two parts: the encoder and the decoder. The encoder reduces the dimensionality of the data into a latent layer representation (the middle layer of the autoencoder that contains the compressed data) which is then reconstructed by the decoder. This takes the latent layer representation as input and outputs a reconstruction of the original input. In the latent layer, the autoencoder learns a representation of the dataset.

The combined use of two autoencoders in our framework aims to detect novelty in systems transaction data. The concept is that autoencoders only classify the most salient features of the class they are trained in. Otherwise, there is a high reconstruction error When presented with data from a different class. By evaluating this reconstruction error with thresholds, novel transactions can be identified.

In the MAE-NAE FRAMEWORK, two autoencoders are used. One autoencoder is trained on normal transactions, while the other is trained on malicious transactions. This enables novelty detection by using reconstruction error to classify novel transactions. If a normal transaction log is passed through the autoencoders, the normal autoencoder should have a low error of reconstruction, whilst the malicious autoencoder should have a high reconstruction error. The opposite is true for a malicious transaction. When a novel transaction is presented to both the normal and malicious autoencoders, the reconstruction errors for both autoencoders will be too similar to classify a transaction as normal or malicious. Mean Squared Error (MSE) is used to measure error because it amplifies large errors, ensuring that errors are distributed across features to maintain a high overall error. This helps reduce the impact of errors in a small number of features by averaging them out.

B. Adjudicator

The job of comparing errors received from the autoencoders is that of the adjudicator. The errors received from the autoencoders can be mapped into a two-dimensional solution space, where the adjudicator is responsible for classifying within this space. The notion of novelty is introduced here by utilising the errors of the autoencoders to identify transactions which it cannot confidently classify.

III. METHODOLOGY

This section presents a methodology for implementing the MAE-NAE FRAMEWORK, an autoencoder-based approach for novelty detection. The goal is to identify novelty within network traffic and classify known data as malicious or normal. Novelty detection is crucial as it allows the classification of transactions that cannot be confidently categorised as malicious or normal, particularly in the presence of concept drift over time. The methodology for generating instances of the MAE-NAE FRAMEWORK is described in the following in terms of training and testing of the models used for the autoencoders and adjudicator.

A. Training

The training of MAE-NAE FRAMEWORK consists of three stages: training of autoencoders, generation of training data for the adjudicator, and train the adjudicator.

Algorithm 1 Training of the MAE-NAE FRAMEWORK

Input: Two set X_m and X_n where X_m is a subset of X containing malicious samples and X_n is a subset of X containing normal samples.

Output: f_m an autoencoder model trained on malicious transactions, f_n an autoencoder model trained on normal transactions, f_a an adjudicator model trained on the MSE from both autoencoders

Split training data 80% into set A and 20% into set B

A_m, A_n, B_m, $B_n \leftarrow$ splitTrainingData(X_m, X_n)

Autoencoder training using set A

$f_m \leftarrow$ autoencoderTraining(A_m)

$f_n \leftarrow$ autoencoderTraining(A_n)

Generate data to train adjudicator

$B \leftarrow B_m \cup B_n$

for all $[b, y] \in B$ **do**

  $c_m \leftarrow$*MeanSquaredError*$(b, f_m(b))$

  $c_n \leftarrow$*MeanSquaredError*$(b, f_n(b))$

  $c \leftarrow [c_m, c_n, y]$

  $C \leftarrow C \cup c$

end for

Adjudicator training

$f_a \leftarrow$ adjudicatorTraining(C)

return f_m, f_n, f_a

Algorithm 1 denotes the overall process of training the MAE-NAE FRAMEWORK. The training data X is split into two sets denoted as A and B. The set A contains 80% of the data from X while B contains the rest. Two autoencoders are trained, one is trained on the malicious samples within set A denoted as A_m while the other is trained on the normal samples of set A denoted as A_n. The resulting autoencoders from training are f_m and f_n for the malicious autoencoder and the normal autoencoder, respectively. Each item in set B is then passed through the autoencoders with the output and input being compared using Mean Squared Error (MSE) producing a new dataset consisting of the value pair (meanSquaredError($f_n(b), b$),meanSquaredError($f_m(b), b$)).

TABLE I
NOTATIONS FOR TRAINING ALGORITHM.

Symbol	Description
X, A, B	A set of transactions and their respective labels, e.g., a set of (x, y) tuples
X_m, A_m, B_m	A subset of the respective set containing malicious transactions and their respective labels
X_n, A_n, B_n	A subset of the respective set containing normal transactions and their respective labels
c_m	The reconstruction error of the malicious autoencoder
c_n	The reconstruction error of the normal autoencoder
y	The label of a transaction, $y \in (malicious, normal)$
c	A tuple of the errors from the autoencoders and label, i.e., (c_m, c_n, y)
C	A set of errors from the two autoencoders, i.e., a set of tuples (c_m, c_n, y)
x, a, b	A transaction from the respective sets, X, A, B
c f_m, f_n, f_a	The malicious autoencoder, normal autoencoder and adjudicator respectively

This new dataset C is then used to train the adjudicator to produce the model f_a.

1) Autoencoders: In the MAE-NAE FRAMEWORK, each autoencoder is trained to reconstruct the provided samples. For example, a generic autoencoder takes an input x, which is then compressed into a latent representation z within the latent layer. Subsequently, this compressed representation is utilised to reconstruct the input x. In the case of MAE-NAE FRAMEWORK, one of the autoencoders is specifically trained to recreate malicious samples, while the other is trained for normal samples. This differentiation arises from the principle that a normal sample should be accurately reconstructed by the normal autoencoder, not by the malicious autoencoder, and vice versa for a malicious sample. Consequently, classification can be performed by comparing the MSE between the input and output for each autoencoder.

2) Adjudicator: Within this framework, a straightforward linear classification method is employed. This technique is depicted in Figure 3, illustrating the errors of each autoencoder along the axes and the linear boundaries showing different classifications. To establish these linear boundaries, percentage thresholds are utilised, as illustrated by the formulas for malicious and normal percentage certainty in Equations (1) and (2) respectively. Adjusting these thresholds alters the gradient of the linear boundaries within the solution space.

To train the adjudicator a grid search is performed on set C to find the optimal values for both the malicious and normal thresholds. Initially, the granularity of the search is at the integer level from 0-100 with promising values from this initial search becoming finer up to a degree of 3 decimal places.

$$Malicious\ Percentage = \frac{NAE\ Error}{NAE\ Error + MAE\ Error} \quad (1)$$

$$Normal\ Percentage = \frac{MAE\ Error}{NAE\ Error + MAE\ Error} \quad (2)$$

TABLE II
DATASET PARTITION FOR KDDCUP99

Data Name	Description
Training set	494,020 data items for training
Testing set	311,029 data items for testing

B. Testing

When the MAE-NAE FRAMEWORK is being tested the following happens. A sample x is passed to both autoencoders to produce $f_m(x)$ and $f_n(x)$ for the malicious autoencoder and the normal autoencoder, respectively. The input and output are compared using mean squared error (MSE) for each autoencoder to produce a reconstruction error. The two errors are then sent to the adjudicator, which applies the two transformations described in Equations (1) and (2). The adjudicator uses these two percentages to classify the transaction according to the function f_a. The structure of the MAE-NAE FRAMEWORK is visualised in Figure 1.

IV. DATASET

The dataset used for evaluation of the MAE-NAE FRAMEWORK is the KDDCUP99[1] dataset. The dataset was generated by simulating a military network environment and capturing the network traffic over a specific time period. It comprises numerous connection records, each representing a sequence of network packets with specific start and end times.

A. Principal Component Analysis

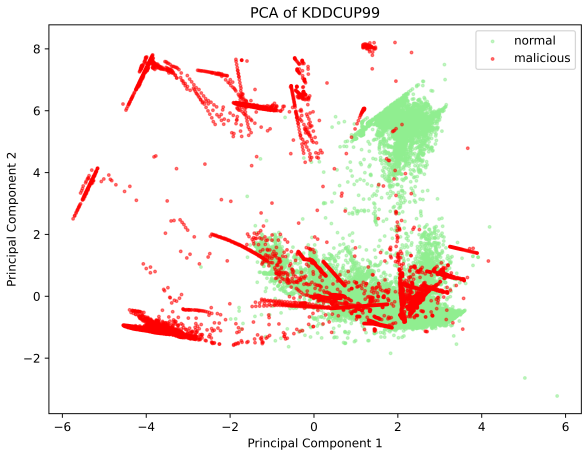

Fig. 2. PCA of KDDCUP99

Principal Component Analysis (PCA) is a technique used to reduce the dimensionality of a dataset while preserving as much variance as possible. This is obtained by finding the vector of maximum variance within the data known as the first principal component. Future principal components are orthogonal to all previous principal components while

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99

maximising variance. PCA can be used as a data exploration technique to expose the internal structure of data in such a way that clearly explains its variance. In Fig.2, two principal components of the training dataset for KDDCUP99 are shown.

It can be observed that there are some clear normal and malicious data clusters, however, in some of the clusters there is a superimposition between normal and malicious data. This overlapping could be removed when looking at more principal components. Nevertheless, it is clear that within the data there are distinct variations between the two. When data items overlap in the data space, it becomes difficult to classify confidently all of them as either normal or malicious. The ability to detect novelty will boost the models' performance by reducing the false negatives and false positives.

V. Experiments

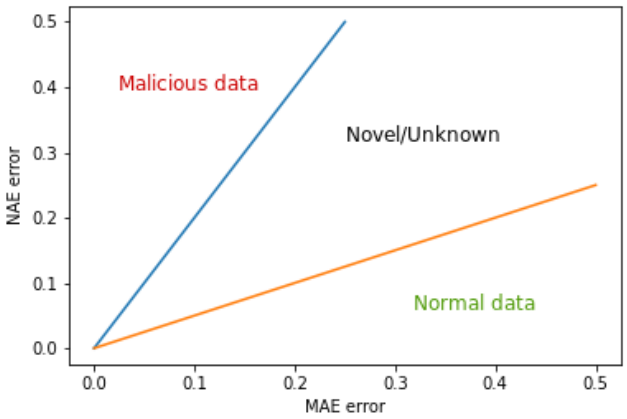

Fig. 3. Demonstration of adjudicator

In this section, the findings of using the MAE-NAE Framework are discussed, which involves: describing the experimental setup, and results of the MAE-NAE Framework.

A. Experimental setup

The computer system used to run all experiments had the following specifications: an Intel® Core™ i7-6500U CPU @ 2.50GHz × 4, running Ubuntu 20.04.4 LTS with a 64-bit OS type and 12 GiB of memory. The experiments were conducted using Python version 3.8.10 along with machine learning modules including SciKit-Learn v1.0.1, TensorFlow v2.6.0, and Keras v2.6.0.

B. Dataset Preprocessing

The preprocessing of the KDDCUP99 dataset involved several key steps to prepare the data for analysis and model training. Firstly, Min-Max scaling was applied to normalise the numerical features, ensuring that all values were rescaled to a range between 0 and 1. This normalisation is crucial for many machine learning algorithms sensitive to the scale of the input data. Secondly, categorical features were transformed using one-hot encoding, converting each categorical value into a binary vector. This step was necessary to handle non-numeric data and allow the machine learning models to interpret the categorical variables appropriately.

TABLE III
Dataset for Training MAE-NAE

Data Name	Description
Autoencoder training set	395,216 data items of KDDCUP99 training set
Adjudicator training set	98,804 data items of KDDCUP99 training set
Testing set	311,029 data items of KDDCUP99 testing set

TABLE IV
Autoencoder Parameters

Setup of training normal/malicious autoencoders		
Autoencoder Topology	78-52-34-23-15-10-6-10-15-23-34-52-78	
Hyperparameters	Optimiser	Adam
	Loss function	Mean Squared Error
	Epochs	150

After the data is preprocessed the training set is split into two using a 80/20 split. This produces three sets as shown in Table III.

C. MAE-NAE Framework

For training the two autoencoders of the MAE-NAE Framework, the autoencoder training set was split into malicious and normal transactions. The malicious transactions are used to train the malicious autoencoder and the normal transactions are used to train the normal autoencoder. The parameters of the autoencoders are outlined in Table IV.

Once the autoencoders are trained the adjudicator training set is passed through the autoencoders and the respective MSE are produced for each of them. This data is then used to train the adjudicator's thresholds using a grid search that gets more granular with each iteration. The respective amounts for each set are shown in Table III.

D. Results

The results of the MAE-NAE Framework show that 94.73% of the testing data is classified as either normal or malicious, leaving 5.27% as novel. The model performs well, with an F1 score of 0.981, which shows its overall effectiveness. Further metrics, such as a precision of 0.991 and recall of 0.971, show that the number of false negatives and false positives is low. These metrics demonstrate the robustness of the MAE-NAE Framework.

The adjudicator classifies transactions based on the errors received by the autoencoders. This can be visualized by plotting the errors received by the NAE on the x-axis and the errors received by the MAE on the y-axis, as shown in Fig.3. Most of the novel data is located near the origin, shown in 4, which indicates that both autencoders are effective at reconstructing them. This highlights the trade-off with using this method as an adjudicator, increasing the model's metrics such as F1, precision and recall, comes at the cost of increased novelty, whereas reducing novelty will adversely affect the model's metrics.

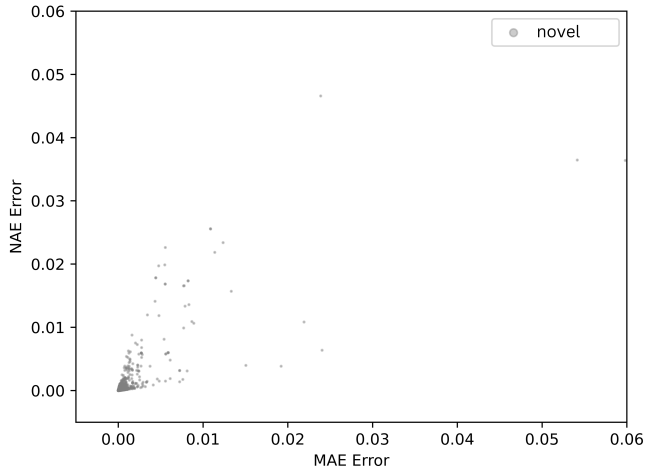

Fig. 4. Demonstration of adjudicator

VI. Evaluation

This section will compare the results presented in this paper to the results of SOTA techniques within the literature used to classify KDDCUP99.

TABLE V
Classification Results of KDDCUP99 for Various Techniques

Technique	F1 Score	Accuracy	Percentage of Novelty (%)
MAE-NAE	**0.981**	**0.969**	5.27
RENOIR[1]	0.958	0.935	0
GC-LSTM[14]	0.910	-	0
CNN[4]	0.89	-	0

The results presented in Table V demonstrate the performance of the MAE-NAE Framework compared to various other techniques from the literature. Our findings indicate that the proposed method achieves higher accuracy than other state-of-the-art techniques. However, a drawback of this approach is the remaining unclassified novel transactions. Despite this, based on comparisons with results from other studies, it is evident that existing methods would have misclassified a significant portion of these transactions.

VII. Related Work

There is a large collection of techniques and frameworks in the literature to identify malicious items, such as malware, infected devices and malicious network traffic. Android malware detection is used frequently due to the large data set of over 130K examples that allows for well-trained models. Many techniques proposed to detect Android malware reported very high F1 scores near 0.99 [3, 8, 11]. At first glance, it looks like this area is solved and methods already exist to detect malware but it was shown that malware changes and models become less effective after 2 years [12]. Another example is the technique proposed by Hafeez et al. [9] which uses fuzzy c-means clustering to analyse IoT network traffic for isolating infected devices. It produces a good model with 0.982 accuracy and is described as a robust solution. There was no mention in the paper of updating the model which would still suffer from concept drift and degrade over time. All of these methods focus on robustness but neglect the resilience aspect.

Autoencoders are widely used to detect malicious or novel examples by looking at the reconstruction error [13]. Autoencoders use thresholds on the inputs to detect whether an input is novel. Ko and Lim [10] go further by detecting malicious inputs using a simple Deep Neural Network (DNN) and an Autoencoder to recreate the explanation map from the DNN. Based on the error, inputs are then classified as normal or malicious. These techniques improve detection by incorporating changes in the input to detect malicious or novel inputs. Further methods update models to dampen the effects of concept drift, such as the method proposed by Barbero et al. [6], which uses a rejection framework that detects data that is moving away from the data the model was trained on. Retraining the model can be triggered by the shift of data, thus making sure the model keeps up with concept drift. This method is not fully autonomous requiring a specialist to analyse rejected data items and label them afterwards. [2] shows a framework called INSOMNIA which uses a DNN to classify network transactions and uses a Nearest Centroid Neighbour classifier to pseudo-label transactions to speed up classification. To keep up with concept drift and remain efficient, it retrains on samples close to the boundaries to avoid wasting computation time. Although this model can handle concept drift, its implementation metrics are not highly effective, with an average F1 score of 0.81.

These detection methods need previous data about the same or similar attacks to detect them, with no notion of novelty. Novel zero-day attacks can pass the detection mechanisms with no problem.

Another detection method relies on monitoring the system and using that to detect changes in behaviour or access permissions. Du et al. [7] uses a method where a model is trained on system logs to detect behaviour that is not normal and may be considered malicious. Suarez-Tangil et al. [15] method monitors the system behaviour by analysing system calls to detect problems with the system to identify malicious attacks. These methods are known to be intrusion tolerant as they detect threats while they happen inside the system. These methods work well when the number of inputs is minimal. When looking at this for a solution to network transactions, the volume is greatly increased, which requires greater computational power.

VIII. Conclusion

This paper has presented the Malicious Autoencoder-Normal Autoencoder Framework (MAE-NAE Framework) for detecting malicious transactions on network traffic while incorporating novelty detection in network transactions. The key motivation behind the MAE-NAE Framework was to handle the ever-changing nature of malicious behaviour, thus providing resilience against attacks. The experiments

have shown that the MAE-NAE FRAMEWORK can detect malicious transactions and at the same time identify novel transactions when there is no confidence in classifying them either as malicious or normal. As part of the evaluation, we compared the MAE-NAE FRAMEWORK against various other state-of-the-art solutions, and it was shown that the proposed approach outperforms them in terms of effectiveness. However, the most significant advantage of MAE-NAE FRAMEWORK against other solutions is its ability to identify novelty, which can be used as a triage mechanism for further classification of novel data.

Future work will focus on the classification of novel transactions. By using a triage-based system, novelty can be further reduced iteratively. Exploring other techniques will help ensure that novel transactions make up an insignificant proportion of the data, making it manageable. Another area of future research will involve evaluating the framework's resilience to adversarial drift, aiming to provide a long-lasting autonomous solution to the problem of detecting malicious network transactions.

REFERENCES

[1] G. Andresini, A. Appice, and D. Malerba, "Autoencoder-based deep metric learning for network intrusion detection," *Information Sciences*, vol. 569, pp. 706–727, 2021.

[2] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: towards concept-drift robustness in network intrusion detection," in *AISec@CCS 2021: Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security, Virtual Event, Republic of Korea, 15 November 2021*, N. Carlini, A. Demontis, and Y. Chen, Eds. ACM, 2021, pp. 111–122.

[3] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: effective and explainable detection of android malware in your pocket," in *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[4] M. Azizjon, A. Jumabek, and W. Kim, "1d cnn based network intrusion detection with normalization on imbalanced data," in *2020 international conference on artificial intelligence in information and communication (ICAIIC)*. IEEE, 2020, pp. 218–224.

[5] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, ser. Proceedings of Machine Learning Research, I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Eds., vol. 27. Bellevue, Washington, USA: PMLR, 02 Jul 2012, pp. 37–49.

[6] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification with conformal evaluation," *CoRR*, vol. abs/2010.03856, 2020.

[7] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. ACM, 2017, pp. 1285–1298.

[8] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. D. McDaniel, "Adversarial examples for malware detection," in *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., vol. 10493. Springer, 2017, pp. 62–79.

[9] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "Iot-keeper: Detecting malicious iot network activity using online traffic analysis at the edge," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 45–59, 2020.

[10] G. Ko and G. Lim, "Unsupervised detection of adversarial examples with model explanations," *CoRR*, vol. abs/2107.10480, 2021.

[11] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristofaro, G. J. Ross, and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models," in *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[12] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 729–746.

[13] A. Rausch, A. M. Sedeh, and M. Zhang, "Autoencoder-based semantic novelty detection: Towards dependable ai-based systems," *Applied Sciences*, vol. 11, no. 21, 2021.

[14] T. Sharma and R. A. H. Khan, "Optimizing network security using lstm algorithm for traffic classification on unswnb15 and kddcup99 dataset," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 8s, p. 671–682, Dec. 2023.

[15] G. Suarez-Tangil, S. K. Dash, P. García-Teodoro, J. Camacho, and L. Cavallaro, "Anomaly-based exploratory analysis and detection of exploits in android mediaserver," *IET Inf. Secur.*, vol. 12, no. 5, pp. 404–413, 2018.