Application of Autoencoder Duets in Anomaly and Intrusion Detection

NITIN MATHUR, University of Cincinnati, United States
DR. CHENGCHENG LI, University of Cincinnati, United States
DR. BILAL GONEN, University of Cincinnati, United States
DR. KI JUNG LEE, University of Cincinnati, United States

Signature-based intrusion detection methods report high accuracy with a low false alarm rate. However, they do not perform well when faced with new or emerging threats. This work focuses on anomaly-based data driven methods to identify potential zero-day-attacks using a specific class of neural networks known as the autoencoder. The significance of this study is that explicit labels are not used in the training process, and the trained model can identify new threats. Our model identified denial-of-service attacks that it had not seen before with a detection accuracy of 95.35 percent.

CCS Concepts: • **Security and Privacy → Intrusion/anomaly detection and malware mitigation**.

Additional Key Words and Phrases: Cybersecurity, neural networks, autoencoders, semi-supervised Learning, CICIDS2017, back propagation, principal component analysis, duets, zero-day-attacks.

ACM Reference Format:
Nitin Mathur, Dr. Chengcheng Li, Dr. Bilal Gonen, and Dr. Ki Jung Lee. 2020. Application of Autoencoder Duets in Anomaly and Intrusion Detection. In *IT Research Symposium '20: School of Information Technology IT Research Symposium, April 14, 2020, Cincinnati, OH*. ACM, New York, NY, USA, 7 pages. https://scholar.uc.edu/

1 INTRODUCTION

Monitoring any complex system or process is done by collecting and analyzing as many metrics as possible. An Intrusion Detection System is used to monitor a computing infrastructure and identify any malicious activity. Malicious activity may include anything that is a threat to the confidentiality, integrity and availability of data. It may range from virus, to unauthorized access and denial-of-service attacks. Modern computing infrastructure can generate several gigabytes of metrics within a short period of time. Real time analysis of these metrics is essential for timely identification of such malicious activity.

1.1 Current Intrusion Detection Methods

1.1.1 Signature Based. The Signature-based method compares incoming metrics with known patterns called signatures. This method is used in commercial products due to its high accuracy and low false alarm rate.

1.1.2 Anomaly Based. This is a data driven method used more often in academic research. An anomaly is a data point that does not conform to the normal or expected behavior of a system [2]. Although not all anomalies are malicious, anomaly detection is an important step towards identifying malicious activity and intrusions. It is also useful in identifying new or emerging threats known as zero-day-attacks.

IT Research Symposium '20, April 14, 2020, Cincinnati, OH

1.2 The Problem

1.2.1 Limitations of Signature Based method. Signature-based systems are perfect for detecting known threats, but they cannot detect new and emerging threats [6]. Also, their Signature databases must be updated regularly.

1.2.2 Limitations of Anomaly Based method. Anomaly-based systems can detect new and emerging threats, but with higher false alarm rates [4]. Most previous work in the literature is based on supervised learning. Experts must label each packet before a classifier is trained. However, the process of acquiring accurately labelled data can be expensive and time consuming. And, just like the signature-databases, labels must be updated regularly too. The researcher must also account for the problem of imbalance in training data.

An anomaly is always unknown. We do not know what are we looking for. This leads us to the first research question.

How can a model identify threats to a computing infrastructure that were never-seen-before with a minimum false-alarm-rate?

1.3 Context of this study

In this work, we investigate the application of representation learning to identify previously known as well as unknown malicious activity without the explicit use of labels. We use a specific kind of a feed forward, non recurrent neural network known as the Autoencoder. We trained an Autoencoder on historical network data that was confirmed to be normal. Our hypothesis is that this trained model can identify data points that deviate from the normal or expected behavior and make them stand out.

1.4 Key findings

An Autoencoder has the potential to model complex non-linear relationships and separate anomalies from normal data. However, there is a significant overlap, and finding an effective threshold is an important consideration which depends on whether 'precision' or 'recall' should have more priority. In this study we use the F1 Score as our metric because it gives equal importance to both precision and recall.

1.5 Significance and Contribution of this work

- No Labels needed.
- New and emerging threats can be identified.
- Low false alarm Rate
- Multiple independent autoencoders can be deployed to cover a wider variety of network data.

2 THEORETICAL FRAMEWORK

An autoencoder is a self-learning neural network where the input and output layers have the same number of nodes and the middle layer has fewer nodes. Autoencoders were first introduced in the 1980s based on work by [7].

Conceptually, an autoencoder resembles an hourglass. Functionally, it consists of an encoder and a decoder as illustrated in Figure 1.

The encoder transforms the unlabeled input data (X) from higher dimensional space to a lower dimensional representation (h). While doing so it eliminates any redundancies, correlated features and features that have minimum variance. This compressed representation retains only the most salient features of the data.

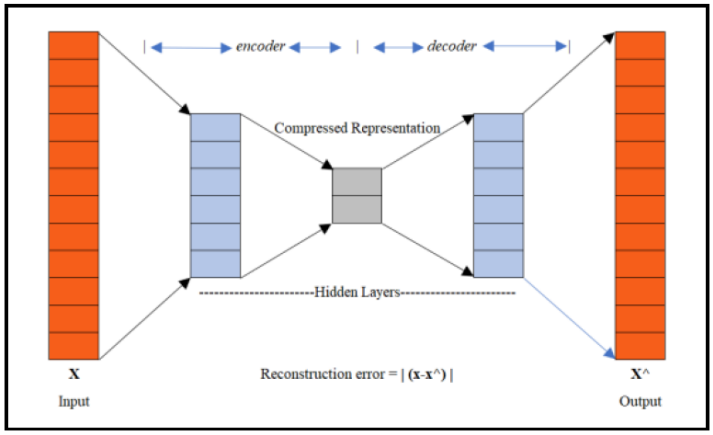


Fig. 1. Autoencoder Illustration

The decoder then reconstructs the original data from this compressed representation This reconstructed data (X') is similar to the original (X), but it is not an exact match. The difference between X and X' is the reconstruction error. This error is measured using a loss function. In this study, the Mean Squared Error (MSE) was used as the loss function. During the training process, the MSE is back propagated through the network after each epoch and the autoencoder updates the weights and biases to minimize the MSE.

The encoder and decoder functions can be represented as follows, where X is the original input, h is the compressed representation, X' is the reconstructed output, w and b are the weights and biases calculated during the training process.

$$h = f(X, w_e, b_e)$$

$$X' = g(h, w_d, b_d)$$

The Mean Squared Error is calculated from the matrices X and X' where n is the number of observations.

$$MSE = 1/n \sum_{i=1}^{n} (X' - X)^2$$

The objective of using autoencoders is to create dimensionally reduced salient features of input data and then to reconstruct them back to their original form with minimum reconstruction error. In this study, an autoencoder is trained on data that is labelled as normal. When new data is passed through the trained model, we record the error of the autoencoder. Our hypothesis is that a high error indicates a deviation from the normal behavior.

3 REVIEW OF THE LITERATURE

Most previous work on Intrusion Detection is based on supervised learning. Autoencoders have primarily been used for feature engineering. One such work by [8] used autoencoders for non linear dimensionality reduction as part of the data preparation process. Very few studies have used Autoencoders as the core component. One such study by [3] achieved a prediction accuracy of 91.7 percent on the NSL-KDD Dataset [10]. The authors added some abnormal data to the training set and defined the classification threshold as a function of the percentage of abnormal data. Another study by [5] simulated a deep autoencoder by daisy chaining four shallow autoencoders. The

shallow autoencoders were trained sequentially. The last layer was a supervised layer that used a SoftMax classifier for the final output. They also implemented back propagation across the entire daisy chain using a greedy layer-wise unsupervised learning algorithm for fine tuning and achieved an accuracy of 94.17 percent on the KDD-99 Dataset. However, one limitation of this model is that each autoencoder must wait for the previous autoencoder in the chain to finish processing. [1] explored the use of a deep autoencoder to detect Denial-of-Service attacks using the recent CICIDS2017 Dataset [9] and achieved an accuracy of 95.73 percent. However, when we analyzed this dataset we found sub-structures within it that can cause random splitting of the data into train and test sets to skew the results if a higher percentage of benign data is collected one sub structure and malicious data from the other sub structure.

4 CICIDS2017 DATASET

4.1 Data Description

The CICIDS2017 dataset [9] is a recent Intrusion Detection evaluation dataset generated by the Canadian Institute of Cybersecurity. It contains five days of network traffic data from Monday to Friday. For this paper we used the data that was collected on Wednesday that contains simulated Denial-of-Service attacks. Each record in this dataset is a 'Traffic Flow' that includes all packets in a single TCP/UDP connection. For instance a TCP flow includes all packets from the first SYN to the last FIN. There are 85 features for each flow.

4.2 Data Analysis

Using Principal Component Analysis to project the dataset on two dimensional space gave us an interesting observation. We found two distinct sub structures within the data as shown in Figure 2. This observation motivated us to train two separate models for each sub structure.

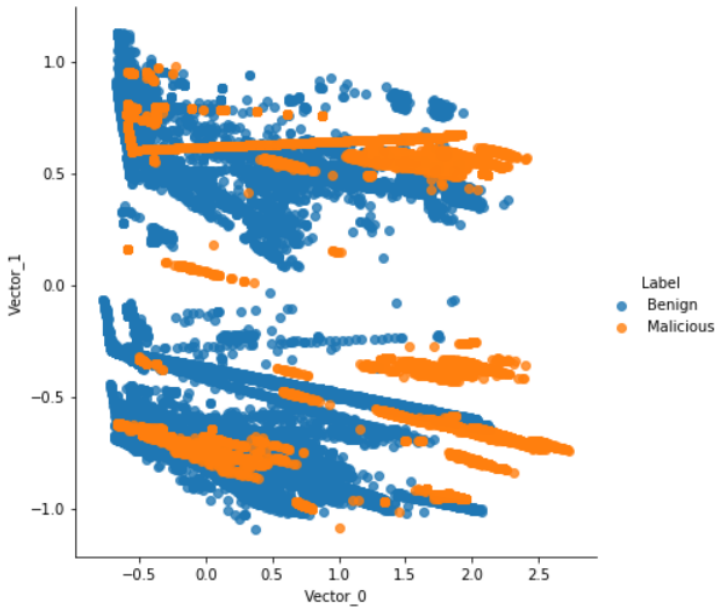


Fig. 2. Data Sub Structures

5 EXPERIMENTS

5.1 Data Preparation

- The features 'Flow ID', Source IP', 'Source Port', Destination IP' and 'Time stamp' removed.
- Infinity and NaN Values were removed
- Labels were made binary - 'Benign (0)' or 'Malicious(1)'. Only 'Benign' traffic was used in the training process.
- Duplicate Records were removed.
- Data was split into a train set (80 percent) and test set (20 percent).
- MaxAbsScaler was used to scale the data. The train data was scaled using the *fitTransform* function. The training parameters were reused to scale the test dataset using the *transform* function. This was done to treat the test data as *new, never-seen-before data.*

5.2 Experimental Setup

Researchers [1] had experimented with different number of hidden layers and neurons. The study concluded that the best performance on this dataset was achieved by an autoencoder with five hidden layers (100, 90, 10, 90 and 100 nodes respectively), batch size =200 and epochs=100. Therefore, we used the same configuration in this study.

We used the Adam optimizer , MSE loss function and the 'reLu' activation function. The rest of the hyper parameters were at their default values.

5.3 Training and Validation

- The train set was was projected on to 2-Dimensional space using Principal Component Analysis and plotted on a scatter plot as shown in Figure 2.
- The original train set created in Section 5.1 was split into two groups. Each data point was moved to either group based on its position within the scatter plot.
- Each group was further split into a train (50 percent), validation (25 percent) and threshold (25 percent) set.
- Two separate models were trained and validated independently using their respective train and validation data. The weights were saved on a local hard drive using a checkpoint.

5.4 Determining the Threshold

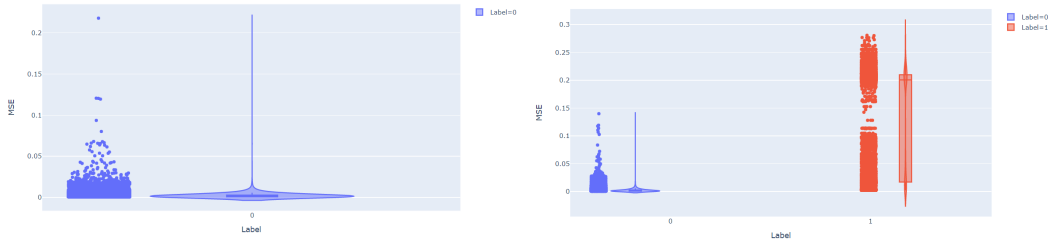
- The threshold sets were input to their respective trained models and the MSE of the autoencoders were recorded.
- The MSE of Normal and Benign traffic were in different ranges, but there was some overlap.
- We used the same method used by [1] for choosing a Threshold that maximized the F1 Score.
- We determined the threshold in two phases. The first phase was used to get an estimate, and the second phase was used to fine tune the threshold as shown in Figure 4.

5.5 Test data

- Finally the test data set was was projected on to 2-Dimensional space. The test data showed the same sub structures, based on which it was split into two groups just like the train data.
- Each test group was passed through its respective trained model and the MSE was recorded.
- Data points that recorded an MSE greater than the determined threshold were classified as malicious using a simple function written in python.

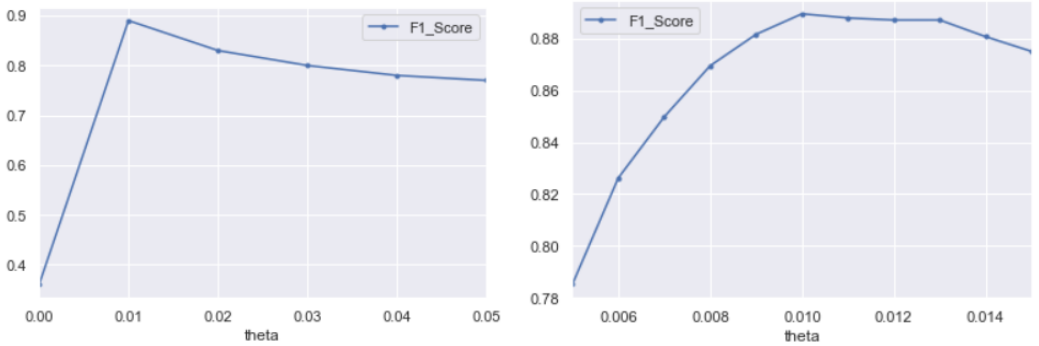
6 RESULTS

The models achieved a consolidated detection accuracy of 95.35 percent with a false alarm rate of 1.49 percent. The MSE Distributions of the validation and threshold subsets are displayed in the violin plots of Figure 3. The threshold MSE that maximized the F1 score was determined to be 0.010. The results are summarized in Table 1.



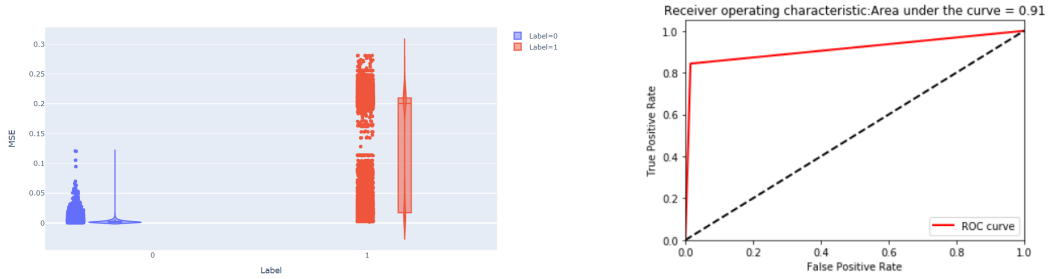
(a) MSE distribution of the 'validation' subset (b) MSE distribution of the 'threshold' subset

Fig. 3. MSE Distribution Violin Plots



(a) F1 Scores (b) Fine Tuned F1 Scores (Max=0.8895 at theta=0.010)

Fig. 4. Finding the optimum F1 score



(a) MSE distribution of the 'test' subset. (b) ROC Curve

Fig. 5. Results

Table 1. Summary of results

Metric	Formula	Value
Precision	TP / (TP+FP)	0.9412
Recall	TP / (TP+FN)	0.8431
Accuracy	(TP+TN) / (TP+TN +FP+FN)	0.9535
False alarm rate	FP / (FP+TN)	0.0149
F1 score	2*(precision * recall)/(precision + recall)	0.8895

7 CONCLUSION, LIMITATIONS AND ONGOING WORK

In this study, we demonstrated the detection of anomalies without using actual labels. We only used historical data that was confirmed to be 'Normal'. Our model achieved a prediction accuracy of 95.35 percent and detected malicious traffic that it had never seen before. The deployment of two models each focusing on a specific pattern within the data has the potential to distribute processing resources since both models work independently.

A limitation of this work is that its validity is restricted to the CICIDS2017 Dataset. A more generalized solution will need further fine tuning. Also, randomized splitting of train and test data could skew the results due to the presence of sub structures. The physical representation of these sub structures is still under investigation and exploring them at a deeper level will be a key to improve the results of this ongoing work.

ACKNOWLEDGMENTS

Although words cannot express the depth of my gratitude, I am eternally grateful to Dr. Chengcheng Li, Dr. Bilal Gonen and Dr. Ki Jung Lee for their guidance that continues to steer this work in the correct direction, and to Dr. Jess Kropczynski for allowing this paper to be presented at the IT Research Symposium 2020.

REFERENCES

[1] Marta Catillo, Massimiliano Rak, and Umberto Villano. 2019. Discovery of DoS attacks by the ZED-IDS anomaly detector. *Journal of High Speed Networks* Preprint (2019), 1–17.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages. https://doi.org/10.1145/1541880.1541882

[3] Hyunseung Choi, Mintae Kim, Gyubok Lee, and Wooju Kim. 2019. Unsupervised learning approach for network intrusion detection system using autoencoders. *The Journal of Supercomputing* 75, 9 (2019), 5597–5621.

[4] Jonathan J Davis and Andrew J Clark. 2011. Data preprocessing for anomaly based network intrusion detection: A review. *computers & security* 30, 6-7 (2011), 353–375.

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

[6] Mrutyunjaya Panda and Manas Ranjan Patra. 2009. Ensemble of classifiers for detecting network intrusion. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*. 510–515.

[7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[8] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. 4–11.

[9] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. In *ICISSP*. 108–116.

[10] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 1–6.