Performance Analysis of Deep-Learning Based Open Set Recognition Algorithms for Network Intrusion Detection Systems

Gaspard Baye

Priscila Silva

Alexandre Broggi

Lance Fiondella

Nathaniel D. Bastian
Army Cyber Institute, U.S. Military Academy, nathaniel.bastian@westpoint.edu

See next page for additional authors

Performance Analysis of Deep-Learning Based Open Set Recognition Algorithms for Network Intrusion Detection Systems

Gaspard Baye*, Priscila Silva*, Alexandre Broggi*, Lance Fiondella*, Nathaniel D. Bastian†, Gokhan Kul*
*University of Massachusetts Dartmouth, Dartmouth, MA
{bgaspard, psilva4, abroggi, lfiondella, gkul}@umassd.edu
†United States Military Academy, West Point, NY
{nathaniel.bastian}@westpoint.edu

NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-7716-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/NOMS56928.2023.10154410

Abstract—**Open Set Recognition (OSR) is the ability of a machine learning (ML) algorithm to classify the known and recognize the unknown. In other words, OSR enables novelty detection in classification algorithms. This broader approach is critical to detect new types of attacks, including zero-days, thereby improving the effectiveness and efficiency of various ML-enabled mission-critical systems, such as cyber-physical, facial recognition, spam filtering, and cyber defense systems such as intrusion detection systems (IDS). In ML algorithms, like deep learning (DL) classifiers, hyperparameters control the learning process; their values affect other model parameters, such as weights and biases, which affect the performance of OSR algorithms. Moreover, OSR introduces additional parameters, making DL classifiers bigger and training them more computationally intensive. Determining the suitable set of hyperparameters and parameters is a computationally expensive task. Alternative OSR algorithms have demonstrated promising results on image datasets, but only limited studies have been performed in the context of IDS. This paper proposes OpenSetPerf, an empirical investigation of three prominent OSR algorithms using a current, real-world network intrusion detection systems (NIDS) benchmark dataset to discover the relationship between the DL-based OSR algorithm's hyperparameter values and their performance. OpenSetperf evaluates these algorithms using quantitative studies with widely used ML performance evaluation metrics.**

I. Introduction

Network Intrusion Detection Systems (NIDS) [1] are one of the most effective lines of defense against attacks designed to steal information, gather cyber-intelligence [2], and tamper with communications [3]. Machine Learning (ML) models are highly effective in detecting cyber-attacks [4] and have therefore been widely adopted [5].

Computer systems generate network packets to transfer information from one computer to another. Various information contained in the packets are used to train ML models to classify the activities performed within the network. However, the majority of ML-based NIDS operate in a closed-world setting [6], meaning that they are tested against classes known at training time and are therefore unable to classify previously unseen examples of classes correctly. The packets generated by malicious actors performing cyber-attacks that the system is not trained to classify, or adversarial actions specifically created to evade ML classifiers make it possible for novel cyber-attacks to remain undetected or their nature misjudged due to this closed-world assumption. Effective detection systems must be capable of adapting in open-world settings, where analysis of unknown traffic patterns is both possible and accurate. Designing NIDS based on open-world scenarios, where the likelihood of unknown network traffic is high, is essential to defend against advanced cyber-attacks.

Open Set Recognition (OSR) techniques [7] have achieved promising results in recognizing unknown inputs in various domains, such as language modeling where OSR could help recognize new sets of text characters [8], [9], new sets of unknown images in computer vision [10], as well as new types of items in object detection [11]. Therefore, they can also be adapted to NIDS [12] to use Artificial Neural Networks (ANN) to monitor network traffic in order to detect, and alert users from novel network attacks. Although deep learning (DL) based NIDS capable of OSR are acknowledged [13] within the research community, there still exist a gap between research and industry adoption. OSR adds an extra layer to DL algorithms, making DL-based NIDS extensive and computationally costly. Also, DL algorithms use hyperparameters [14] to control their learning process, which poses the challenge of selecting the optimal range of hyperparameters values that achieve favorable results under different scenarios.

This paper conducts an in-depth evaluation of prominent DL-based OSR algorithms and examines the impact of factors such as hyperparameters on the performance of these algorithms. The work constructs a DL-model with OSR capabilities and evaluates the performance of hyperparameter needs for various OSR approaches. We modify the deep neural network architecture's end layer to incorporate the various OSR implementations. This modification ensures that all OSR algorithms are evaluated on the same inputs, base neural network, and hyperparameters. This approach also modifies the traditional single architecture neural network to incorporate a multi-architecture evaluation technique comprised of a 1D convolutional neural network and a fully-connected neural network. In doing so, the various OSR algorithms in the final layer can be evaluated on multiple architectures, while maintaining the same hyperparameters.

The specific contribution of this paper is that we show the impact of hyperparameters on the performance of OSR algorithms, enabling the identification of the hyperparameters that are more important and significantly affect the performance of DL-based OSR algorithms.

II. Open Set Recognition

Open Set Recognition algorithms are ML algorithms capable of recognizing the unknown. Open sets refer to how we wish to approach classification problems. As a simple example, assume that we have a traditional multi-class classifier that classifies network intrusion attacks as either Denial of service (DOS) attacks, Web attacks, or Secure Shell (SSH) brute-force attacks. If an attacker performs an SSH brute-force attack, the multi-class classifier can classify this traffic as SSH Bruteforce. Any traffic fed to the multi-class classifier will only fall between those closed sets of three classes previously mentioned due to the closed set nature of traditional ML models (See Figure 1-A). On the other hand, when we give the classifier traffic related to a buffer overflow attack, which is an attack aiming to run malicious codes to gain unauthorized access to corporate systems, it will not belong to any of the three classes. Despite not belonging to any classes, the model still classifies this traffic as a DOS or a Web attack, therefore, the classifier will be faulty. Ideally, an OSR algorithm will enable this model to label the traffic as unknown, as it is not part of the trained classes. OSR modifies the output layer of the deep neural network to allow the prediction of unknown classes (Figure 1-B).

Opening a machine learning algorithm to involve unknowns introduces additional issues. For example, unbounded open space problem [6] creates infinite possibilities of input space via which classification input can arise.

A. Unbounded Open Space Risk

An Open Space is an area far from the training examples. A traditional classifier without open space awareness is bound to the open space risk and will classify everything, including unknown items "???". We observe an open space risk when input data found anywhere within a particular open space are classified as one of the defined known classes even if they don't necessarily belong to that class. Classification algorithms always classify inputs into one of their predefined classes. Since there is no boundary to where the input vectors need to belong to be classified under a specific class, the input space continues forever. Let f be a measurable function that can recognize a set of inputs $X \in \{X_1, X_2, ..., X_N\}$ under known classes $\hat{C}$. Let $U_{\hat{C}}$ be the union of all balls with radius r having all training data for all known input examples $x \in \hat{C}$, and let O be a certain open space where $O \subseteq X - U_{\hat{C}}$. The set of class $\hat{C}$ with an open space risk $O_R(f)$ is represented as :

$$O_R(f) = \frac{\int_O f_C(x)\,\mathrm{d}x}{\int_{U_{\hat{C}}} f_{\hat{C}}\,\mathrm{d}x} \tag{1}$$

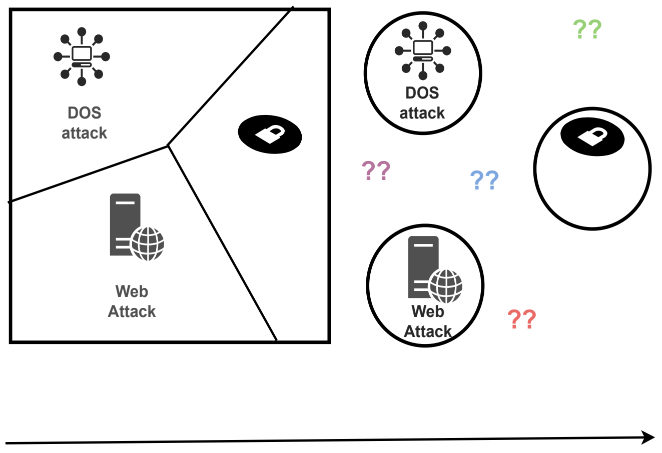

Fig. 1. Multi-class classification of both open set and closed set recognition using NIDS benchmark data-set

where open space risk can be defined as a measure of positively labeled open space relative to an overall measure of positively labeled space.

The unbounded nature of the open space risk problem can be solved using the compact abating probability model (CAP). We threshold the CAP to create a bounded open space. CAP models demonstrated value in novelty, anomaly, and outlier detection [6].

B. Non-OSR Algorithms

Machine learning is a branch of artificial intelligence (AI) that uses data and algorithms to mimic how the human brain learns and recognizes items. Deep learning is a particular form of machine learning in which an algorithm with multiple layers of artificial neurons learns on a vast dataset. An artificial neuron is a mathematical function that simulates the behavior of biological neurons, a neural network. An artificial neural network consists of these artificial neurons. When we add multiple layers of neural networks, we build a deep neural network. The output layer is a layer of the neural network that makes the final decision. It often does this by an algorithm called Softmax, representing the final layer that determines whether an input belongs to a specific class or not. The typical way Softmax accomplishes this is by normalizing the output of a deep neural network logits, or the final layer before the activation function is applied to a probability distribution over a set of predicted output classes. The Softmax output layer determines the confidence of an output specifically from the Softmax score, where a higher score indicates greater confidence of belonging to the class.

Let the output of a Softmax be the vector $\vec{V}$, which represents the probability of each possible outcome or class; the vector $\vec{x}$ represents the input to a Softmax function S_f. $\vec{x_i}$ is the i^th element of the input vector with values between $-\infty$ and $+\infty$ and N is the number of possible outcomes or classes. Mathematically, Softmax is represented as,

$$S_f(\vec{x_i}) = \frac{e^{\vec{x_i}}}{\sum_{k=1}^{n} e^{\vec{y_j}}} \quad (2)$$

where $\sum e^{\vec{y_j}}$ is a normalization term that ensures that values from the output vector $S_f(\vec{x_i})$ sum to 1 for the i^th class and that each i^{th} class is in the range 0 and 1, making up a proper probability distribution.

Softmax has infinite open set risk or infinite area, where something can be classified as one of the known classes. It has this risk because it is still just a set of lines with an infinite area on either side. Since Softmax does not solve the open space risk problem, DL algorithms using softmax are not considered to be OSR algorithms. However, there are numerous algorithms that solve the open space risk problems in various ways.

C. OSR Algorithms

1) OpenMax : The OpenMax concept first appeared in Abendale et al. [15], where the authors present a method for adapting deep networks for open set recognition by introducing a new model layer, OpenMax, which estimates the probability that an input belongs to an unknown class. A critical element of estimating the unknown probability is adapting Meta-Recognition concepts to the penultimate layer of the network's activation patterns. OpenMax reduces the number of obvious errors made by a deep network by rejecting "fooling" and unrelated images. An open set recognition solution is provided formally by the OpenMax concept, which achieves bounded open space risks.

OpenMax is an improvement of Softmax, enabling it to recognize unknown inputs. Assume we have a set of classes $i = 1, 2, 3, ..., N$ possessing multiple data points in which each class is represented as a point, a mean activation vector (MAV) with the mean μ_i, computed over only the correctly classified training example. They measure the distance between the MAV and an input sample. The lack of uniformity in the activation vector (AV) for different classes presents a greater challenge; hence, OpenMax uses a per-class meta-recognition model. OpenMax uses libMR [16] library to provides core MetaRecognition and Weibull fitting functionality, FitHigh. MetaRecongition is the ability of a machine learning algorithm to be aware of its learning process. The Weibull fitting function, in the form of Weibull cumulative distribution function (CDF) probability, $W(t) = 1 - e^{-(\frac{t}{\lambda})^k}$, where λ and k are the scale and shape parameters respectively. The Weibull function provides a flexible way to detect outliers in the input data by fitting on the largest distances between all correct positive training instances and the associated μ_i. This results in a parameter ρ_i, a simple rejection model used to estimate the probability of input being an outlier with respect to class i.

2) Energy-based Out-of-distribution Detection: Weitang et al. [17] proposed an energy-based framework for out-of-distribution (OOD) detection, demonstrating that energy scores distinguish in- and out-of-distribution samples better than traditional softmax scores. Energy scores are less sensitive to overconfidence than softmax confidence scores because they are theoretically aligned with the probability density of the inputs. Using this framework, energy can be flexibly used as an objective scoring function for pre-trained neural classifiers and as a trainable cost function to shape an energy surface for OOD detection.

To understand how OOD works, consider a discriminative neural classifier $f(x) : R^D \to R^K$, where an input $x \in R^D$ maps K real-valued digits known as logits. The softmax function, $P(y|x) = \frac{e^{f_y(x)/T}}{\sum_i^k e^{f_i(x)/T}}$ uses logits to derive a categorical distribution. Where $f_y(x)$ denotes the y^{th} index of $f(x)$, i.e. the logit for the y^{th} class label and T is the temperature parameter. For a given input x, we can define the free energy $E(x, f)$ over $x \in R^D$ parameterized by the neural network $f(x)$ as follows: $E(x; f) = -T.log \sum_i^k e^{f_i(x)/T}$

III. Performance Analysis of Deep-Learning Based OpenSet Recognition algorithms

In this section, we define our approach to evaluate the performance of OSR algorithms on NIDS benchmark datasets, we detail the experimental setup, the different evaluation metrics used to measure the performance of the different algorithms, the dataset used, and the techniques used to fix data imbalances.

A. Evaluation Architecture

We propose an architecture which evaluates various OSR algorithms while maintaining the various hyperparameters. This is done by modifying the traditional single architecture neural network to incorporate a multi-architecture evaluation technique comprising a 1D convolutional neural network and a fully-connected neural network. By doing so, the various OSR algorithms at the end layer can be evaluated on multiple architectures while preserving the same hyperparameters. In order to maintain a single configuration for all OSR algorithms, we create a configuration file which can be tuned to adapt with the variations in hyperparameter settings. Figure 3 shows a simplified representation of the architecture used for our test. The figure contains an input, which is the network intrusion detection dataset fed to the DL-algorithm. After the input layer, the neural network possesses $N-1$ layers, and the Nth layer contains the OSR algorithms. The predicted outputs for the various algorithms are collected and evaluated. Our code and reproducibility instructions are open-sourced [1].

An OSR algorithm classifies the input NIDS benchmark dataset into either known or unknown attack classes. Evaluation metrics are used to assess the algorithm's ability to detect unknown inputs. According to Fairuz et al. [18], accuracy, precision, false alarm rate (FAR), and recall are the most widely used metrics for evaluating ML models. In this study, accuracy, precision, recall, as well as F1-score are calculated for the various OSR algorithms. A single metric cannot objectively measure [19] an algorithm. For instance, consider the difference between accuracy and false positives (FP). A good accuracy rating is based on the number of correctly classified

[1]https://github.com/bayegaspard/OpenSetPerf.git

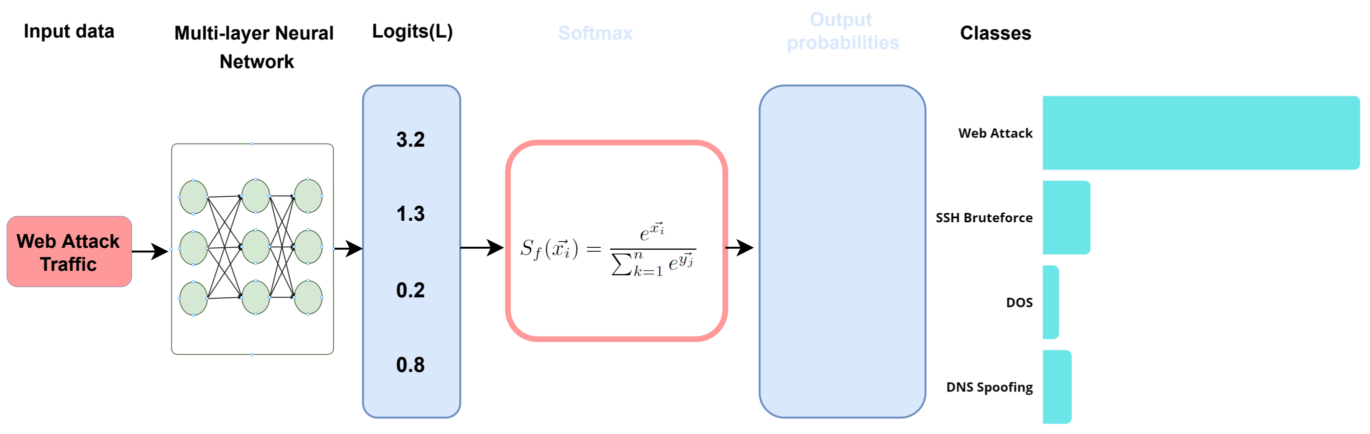

Fig. 2. Classification of network intrusion detection system traffic using Softmax

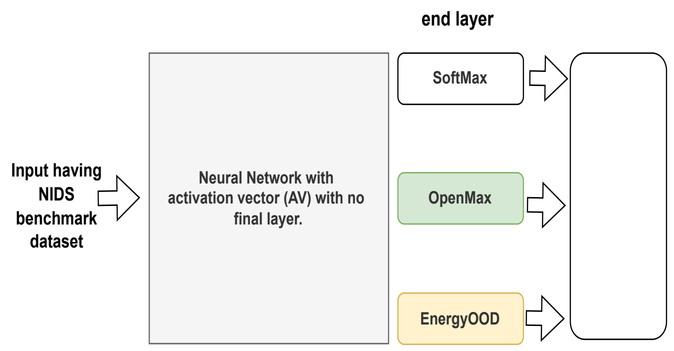
end layer
Input having NIDS benchmark dataset
Neural Network with activation vector (AV) with no final layer.
SoftMax
OpenMax
EnergyOOD

Fig. 3. Evaluation architecture for DL based OSR algorithms using NIDS dataset

samples, while a good FP rating is based on the number of false alarms. Imagine a dataset where most examples fall into the positive category, and only a few fall into the negative category. Naive classifiers that classify all samples as positive would have a low FP but a good accuracy score.

1) Recall and Precision: Equation 3 represents the Recall, which is the proportion of all true positive (TP) classifications compared to the sum of all true positives and false negatives (FN). For NIDS, it is the proportion of samples correctly classified as malignant to those that are malignant [1]. Precision [1], equation 4, is the ratio describing the correct predictions, TP with respect to all positive predictions, TP and False Positive (FP).

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

2) $F_1 - score$: $F_1 - score$ [20] of a model is defined as the harmonic mean between its precision and recall. The $F_1 - score$ is particularly useful when there is a heavy class imbalance and when false positives and false negatives have different costs. For example, when dealing with imbalanced data, the $F_1 - score$ is a more accurate metric than accuracy. As a rule of thumb, an $F_1 - score$ of 1.0 indicates a successful classifier, while 0 indicates a failed one.

$$F_1 - score = 2 \times \frac{Precision \times recall}{precision + recall} \tag{5}$$

3) Accuracy: Accuracy measures how many samples were correctly classified compared to how many samples were totaled. It is only appropriate to use the metric when the dataset is balanced. Leevy et al. [21] and Ahmad et al. [22] do not recommend using accuracy for NIDS due to their imbalanced nature.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

B. Dataset

This work uses a CSV (comma separated values) dataset from the popular NIDS benchmark PCAP (packet capture) format CIC (Canadian Institute of Cybersecurity) IDS 2017 [23] dataset. Generally, network traffic is captured and stored using PCAP files. The CSV dataset is generated using the Payload Byte [24] tool. Unlike other packet extraction tools, this tool enables the extraction and labeling of network packet features, including the payloads. A payload is the section of a network packet that carries the data during communication. Attackers often use this section to hide malware or malicious code. Capturing information about the payload is necessary to create a complete real-world NIDS dataset.

The dataset contains $1,410,255$ datapoints under 15 different classes. Each class is composed of multiple packets, all classified under benign and attack network traffic. Each packet has eight features, subdivided into packet header and packet data. The packet header contains information used by the destination computer to decode the packet and extract the payload. The packet header includes source IP (internet protocol), destination IP, source port, destination port, and protocol (user data protocol or transmission control protocol). The packet data carries the payload, which is the data, represented in bytes transmitted over the network.

IV. Results

This section illustrates the performance of three prevalent OSR algorithms on the NIDS benchmark dataset. The test was conducted on various hyperparameters. From the conducted experiments, we realized that some hyperparamters such as the learning rates, batch size and percentage of unknowns impacts the various OSR algorithms meanwhile the number of epochs have limited effect on OSR algorithms.

Fig 4 shows a graph of the number of epochs versus performance shown as $F1-score$, recall, precision, and accuracy. The number of epochs represents the number of rounds the algorithms receive training data. As we vary the number of epochs (10, 50, 100), the performance of Softmax increases but those of Energy and OpenMax are not affected sifnificantly. In every round, OSR algorithms only detect unknown patterns rather than trying to have better outcomes per more rounds. Hence, more rounds still facilitate fairly constant the detection of unknowns.

Fig 5 shows the variations of learning rate versus the performance captured as $F1-score$, recall, precision, and accuracy. The learning rate determines the algorithm's learning step size. As we vary the learning rate (0.01, 0.05, 0.1), we can see that the F1-Score of all three algorithms decrease. Even though we have many performance measures for the unbalanced dataset, F1-score is often the best performance indicator in some instances because it captures the bigger picture. The decrease is because the algorithm starts taking wider steps and fails to converge at the local minimum.

The batch size is the number of samples processed before a model update. Fig 6 represents the variation of batch sizes versus the algorithm's performance, such as $F1-score$, recall, precision, and accuracy, exhibiting slight changes in performance as we vary the size of batches from 10, 50, 100. This is primarily because as the size of data changes, the number of unknown data per batch also slightly changes, leading to a change in performance. In this graph, Sofmax shows some performance impacts due to its ability to classify most data samples within a batch.

Fig 7 shows the percentage of unknowns versus the performance, represented as $F1-score$, recall, precision, and accuracy. The graph shows an increase in the percentage of unknowns, i.e., the number of classes not present at training time, from 26%, 40%, and 66% leading to an increase in the performance of OSR algorithms. However, we can also see that Softmax performance reduces as we increase the percentage of unknowns. This is because Softmax is not inherently designed to understand unknowns, but OSR algorithms are developed to identify unknowns.

V. Conclusion and Future Work

In this work, we proposed an architecture to evaluate DL-based OSR algorithms on the NIDS, determining which hyperparameters had the highest impact. As our experimentation is ongoing, we have observed the following trends in our preliminary results. Energy-based OOD and OpenMax demonstrate superior capabilities in identifying unknowns when

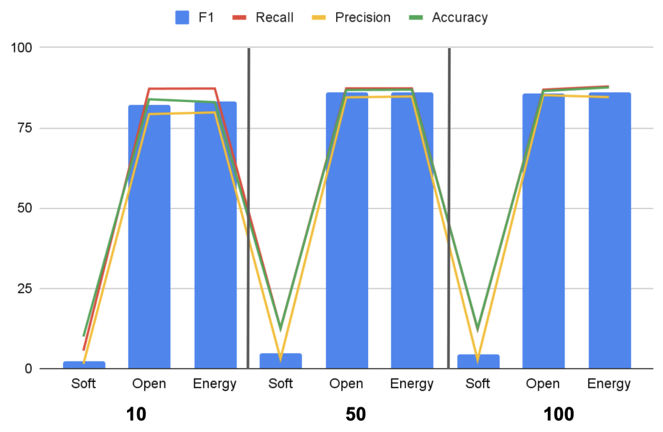

Fig. 4. Number of epochs

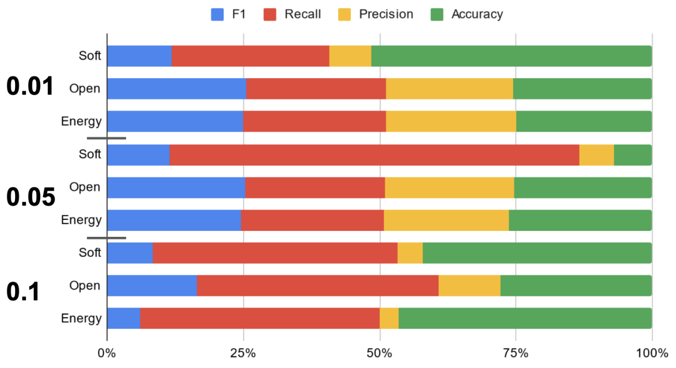

Fig. 5. Learning rates

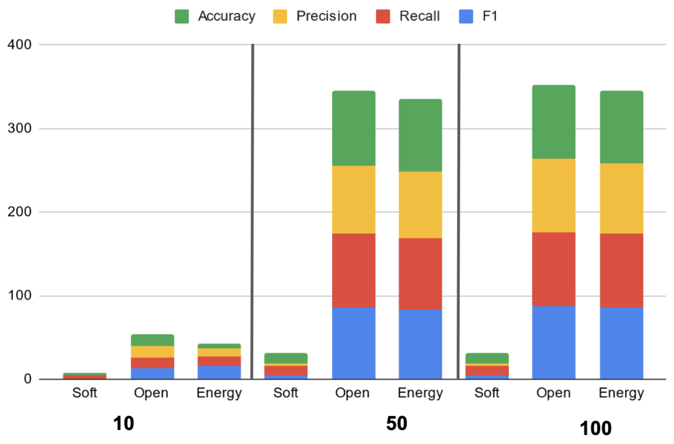

Fig. 6. Batch size

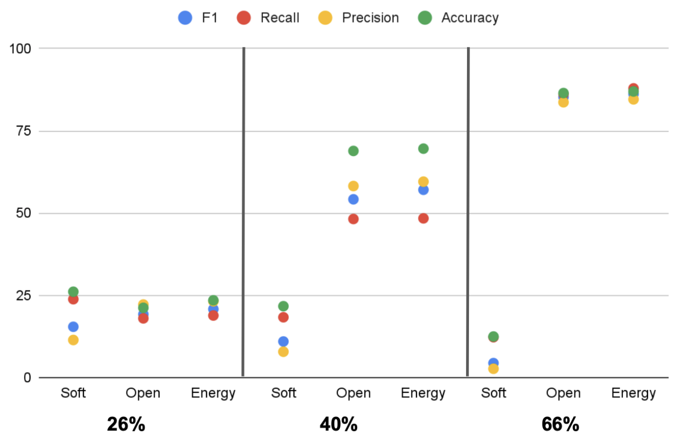

Fig. 7. Percentages of Unknonws

compared to SoftMax. Notably, Energy-based outperforms both OpenMax and Softmax with regards to the percentage of unknowns, while OpenMax exhibits better performance with larger batch sizes. Conversely, Softmax performs best with a lower percentage of unknowns as expected. Furthermore, our analysis suggests that OpenMax and Energy have similar impacts on learning rates.

In future work, we plan to conduct more detailed studies on the performance of various OSR algorithms in order to be able to increase resilience of NIDS in the presence of unknown network traffic. Automated hyperparameter tuning and on-the-go algorithm selection based on experienced network traffic when the NIDS performance start to drop will contribute to the research in the field of resilience of NIDS. We will further perform research on generative models to identify strengths and weaknesses of resilient NIDS.

Acknowledgment

This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0160, the U.S. Army Combat Capabilities Development Command (DEVCOM) C5ISR Center under Support Agreement No. USMA21056, and the U.S. Army DEVCOM Army Research Laboratory under Support Agreement No. USMA 21050. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The U.S. Government reserves a royalty-free, nonexclusive and irrevocable right to reproduce, publish, or otherwise use this data for Federal purposes, and to authorize others to do so in accordance with 2 CFR 200.315(b).

References

[1] G. Baye, F. Hussain, A. Oracevic, R. Hussain, and S. A. Kazmi, "Api security in large enterprises: Leveraging machine learning for anomaly detection," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2021, pp. 1–6.

[2] R. Syed, "Cybersecurity vulnerability management: A conceptual ontology and cyber intelligence alert system," *Information & Management*, vol. 57, no. 6, p. 103334, 2020.

[3] D. Liu, D. Liang, and C. Wang, "A novel three-way decision model based on incomplete information system," *Knowledge-Based Systems*, vol. 91, pp. 32–45, 2016.

[4] S. Chesney, K. Roy, and S. Khorsandroo, "Machine learning algorithms for preventing iot cybersecurity attacks," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2020, pp. 679–686.

[5] R. Prasad and V. Rohokale, "Artificial intelligence and machine learning in cyber security," in *Cyber Security: The Lifeline of Information and Communication Technology*. Springer, 2020, pp. 231–247.

[6] T. E. Boult, S. Cruz, A. R. Dhamija, M. Gunther, J. Henrydoss, and W. J. Scheirer, "Learning and the unknown: Surveying steps toward open world recognition," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 9801–9807.

[7] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1757–1772, 2012.

[8] V. M. Venkataram, *Open set text classification using neural networks*. University of Colorado Colorado Springs, 2018.

[9] L. Shu, H. Xu, and B. Liu, "Doc: Deep open classification of text documents," *arXiv preprint arXiv:1709.08716*, 2017.

[10] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[11] Z. Ni and B. Huang, "Open-set human identification based on gait radar micro-doppler signatures," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8226–8233, 2021.

[12] H. Lindstedt, "Methods for network intrusion detection: Evaluating rule-based methods and machine learning models on the cic-ids2017 dataset," 2022.

[13] C. Wang, B. Wang, Y. Sun, Y. Wei, K. Wang, H. Zhang, and H. Liu, "Intrusion detection for industrial control systems based on open set artificial neural network," *Security and Communication Networks*, vol. 2021, 2021.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[15] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[16] W. J. Scheirer, A. Rocha, R. Michaels, and T. E. Boult, "Meta-recognition: The theory and practice of recognition score analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, pp. 1689–1695, 2011.

[17] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," *Advances in Neural Information Processing Systems*, 2020.

[18] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.

[19] A. Botchkarev, "Evaluating performance of regression machine learning models using multiple error metrics in azure machine learning studio," *Available at SSRN 3177507*, 2018.

[20] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[21] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, 2020.

[22] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[24] Y. A. Farrukh, I. Khan, S. Wali, D. Bierbrauer, J. A. Pavlik, and N. D. Bastian, "Payload-Byte: A Tool for Extracting and Labeling Packet Capture Files of Modern Network Intrusion Detection Datasets," *Proceedings of the 9th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT2022)*, 12 2022.