Low Complexity Shallow Neural Network With Improved False Negative Rate for Cyber Intrusion Detection Systems

Jörg Ehmer, Bertrand Granado, Julien Denoulet, Yvon Savaria, Jean-Pierre David

Low Complexity Shallow Neural Network With Improved False Negative Rate for Cyber Intrusion Detection Systems

Jörg Ehmer*, Bertrand Granado*, Julien Denoulet*, Yvon Savaria† and Jean-Pierre David†
*Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
†Electrical Engineering Department, Ecole Polytechnique, Montréal, Québec, Canada

Abstract—**Economic value creation increasingly takes place online or is tightly coupled to some kind of online service. At the same time, malicious network activities are causing growing losses in the strongly digitalized economies. Hence, protecting network communication infrastructure is an important challenge for companies and public institutions alike. Machine learning algorithm based network intrusion detection systems (NIDS) are often used to detect sophisticated attack patterns. However, the detection quality of those algorithms suffers greatly from the imbalanced nature of network flow data. Undetected attacks can cause great damage, so it is essential that a NIDS performs at its best in order to detect as many attacks as possible. In our article, we propose an improved loss function in order to reduce the number of false negatives produced by an artificial neural network (ANN). Based on the CIC-IDS17 dataset, we show that our proposed algorithm running on a shallow neural network (single layer with 110 neurons) successfully classifies a variety of recent network attacks with a F1-score above 99%.**

I. Introduction

Detecting harmful activities in a network's flow of packets is algorithmically and computationally challenging. Traditional network intrusion detection systems (NIDS) often use known patterns to identify fraudulent behaviour. Those patterns have to be carefully crafted in order to achieve acceptable detection rates. Attackers, however, continuously adapt their strategies in order to avoid detection. That is why security professionals and researchers investigate new methods to deal with the ever-growing number of threats present in current day computer networks. One of these new emerging methods is the use of machine learning. Rapid developments in the field of artificial intelligence and deep learning have led to a wide variety of algorithms capable of detecting anomalies in the flow of network traffic and even classify certain attacks present within those network flows. Proposed algorithms in the literature include recurrent neural networks (RNNs) [8], convolutional neural networks (CNNs) [12] [7], support-vector machines (SVNs) [17] and broad learning systems (BLSs) [13] [9]. An additional challenge, that all of those approaches have in common, is the fact that network attacks are ordinarily rather rare events and that undetected attacks can lead to great costs. The number of those attacks that were misclassified as benign network traffic is called false negative rate in the context of deep learning. To improve the security of a communication network it is thus important to minimize this kind of error. False positives, which are benign network flows that were classified as an attack are somewhat less important as they can be dealt with at a later stage of the intrusion detection process.

A. Imbalanced learning

Training neural networks to detect rare events is a challenging task. Strongly imbalanced training data can cause a network to classify almost all entries as part of the majority class, the class containing the most samples. In such a case, the accuracy performance measure might be relatively good, even if the actual classifications for samples of the minority classes are in fact quite poor.

B. Strategies to deal with imbalanced datasets

Guo Haixiang et al. [6] provide in their article a comprehensive overview of several possible strategies that deal with imbalanced data. One such strategy is to change the training dataset via resampling into one that is more balanced. This is mainly achieved by oversampling [4] or undersampling [2]. Oversampling means that samples of the minority class are copied to increase the number of such data points. However the repeated use of the same data points can cause the network to overfit. A way to prevent overfitting is to generate new data points from the minority classes as it is proposed in the Synthetic Minority Oversampling Technique (SMOTE) and its variants [1] [11]. Another technique is to use generative adversary networks (GANs) to generate additional samples for the minority class as proposed in [14]. Leaving out some of the majority classes data samples constitutes the undersampling strategy. This procedure effectively shrinks the majority sample set and leads to a more balanced training dataset. However, this can lead to inferior classification results of the majority class due to a less diverse training set. The Difficult Set Sampling Technique (DSSTE) proposed by Liu et al. [16] is an algorithm that combines over and undersampling in order to decrease the number of majority data points, while increasing the number of minority samples. Ensemble methods and cost-sensitive learning represent other strategies that deal with imbalanced data. Cost-sensitive learning incorporates the imbalance of the training data into the training process by associating an additional cost to the imbalance of the training data. Such methods often require a deep understanding of the impact of the class imbalance in order to properly calculate its associated cost. Guo Haixiang et al. [6] mention in their article that the difficulty to acquire sufficient knowledge about the cost in relation to the class imbalance greatly inhibits the widespread use of cost-sensitive learning. Recent developments in machine learning, however, have shown that it is possible to achieve cost-sensitive learning without the need for specially crafted cost matrices by using loss functions such as the attack-sharing loss proposed by Dong et al. [15].

C. *Imbalanced data classification domains*

Imbalanced learning is a problem present in major fields of research or engineering. A variety of application domains can be found in [6]. The focus of this article lies in the information technology domain and particularly in its network security aspects. The steadily growing spread of IoT devices and the ongoing shift towards a more digital economy emphasize the need to secure our communication networks. Most of the network's traffic is of benign origin, attacks, on the other hand, are normally rare events. It is this biased distribution of benign versus malicious traffic that places the task of securing a computer network in the domain of imbalanced learning.

D. *Aim and organization of the paper*

In this article we propose an improved attack-sharing loss function capable of reducing the false negative rate for the task of network attack classification. Furthermore we propose a neural architecture that achieves good detection results while being sufficiently lean in order to be deployed in an embedded environment. In Section II we will describe the training dataset used. Section III will deal with the changes we propose to further improve the loss function. Our implementation will be described in Section IV. In Section V we will present our results, followed by a comparison of the two loss functions in Section VI which will be succeeded in Section VII by our conclusion.

II. Analysis of the problem and prior work

In the context of this article, we used the publicly available dataset CIC-IDS17 [18] from the Canadian Institute for Cybersecurity to train a neural network capable of detecting attacks within a computer network. The 2.83 million samples containing data set comprises 78 network flow features and a class label describing whether the flow belongs to one of 14 network attacks or a benign user interaction. A full description of the data set can be found in [10]. Since the provided data was heavily skewed towards the benign class, we decided to combine attacks of a common family and excluded those with insufficient data points as it was proposed by [15]. The resulting simplified sample set provided in Table I remains obviously quite imbalanced. To address this problem, Dong et al. [15] proposed an adaptation of the cross-entropy loss function, which they called attack-sharing loss that can be quantified with equation 1.

TABLE I: Distribution of classes in the simplified dataset

Class	Percentage	Count
BENIGN	80.42	2 273 097
DoS	8.94	252 661
Infiltration	5.62	158 966
DDoS	4.53	128 027
Brute-Force	0.49	13 835

$$J_{AS} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N} \lambda \Big(I(y^{(i)},1) log\ p_1^{(i)} + \sum_{j=2}^{c} I(y^{(i)},j)\ log\ (1-p_1^{(i)}) \Big)\right] \quad (1)$$

J_{CE} is the vanilla cross-entropy loss function, N is the number of training samples in the batches, c represents the number of classes in the training dataset. The factor λ scales the effect of the regularizing term. A small value of λ makes the loss function behave almost like the vanilla cross-entropy function. $y^{(i)}$ is the i-th label and $p_1^{(i)}$ the i-th predicted probability for the first class (the majority class). The function $I(a,b)$ is the indicator function which is defined by equation 2.

$$I(a,b) = \begin{cases} 1 & if\, a = b \\ 0 & otherwise \end{cases} \quad (2)$$

III. Proposed solution

In order to further tune the regularization term of the original attack-sharing loss function, we introduce two new parameters α and β. Our aim with this new loss function (3) is to reduce the false negative rate even more by increasing the regularization penalty contribution in the case where the current sample belongs to the minority class.

$$J_{AS} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N} \left(\alpha\ \Big(I(y^{(i)},1) log\ p_1^{(i)} \Big) + \sum_{j=2}^{c} s_j\ I(y^{(i)},j)\ log\ (1-p_1^{(i)}) \right)\right] \quad (3)$$

The factor α scales the penalty contribution for the case where the current training sample belongs to the majority class. The second parameter β is multiplied by the inverse frequency of the current samples minority class, resulting in the scaling factor s_j as shown in (4).

$$s_j = \beta * (1 - \frac{n_j}{N_{mc}}) \quad (4)$$

In equation (4) n_j represents the number of samples in the current batch belonging to the minority class j, N_{mc} represents the total number of minority samples in the batch. The scaling factor s_j thus reflects the distribution of the minority classes in the current batch. Rare samples in the batch cause therefore a higher regularization penalty which in turn causes a stronger displacement of the decision boundary towards the attack classes. The two factors α and β are hyperparameters which allow a more nuanced tuning of the training process. In our tests for which detailed results are reported later, we achieved good results with an empirical value of 5 for α and 20 for β.

TABLE II: Recall (Rec) and precision (Pre) metrics for the individual classes, (a) reference results for a 10 layer neural network published in [15], (b) original and (c) improved loss functions used with our proposed single layer neural network

	BENIGN		Brute-Force		DDoS		DoS		Infiltration	
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
Original J_{AS} [15] (a)	0.8850	0.9406	0.8290	0.4100	0.7631	0.8319	0.8877	0.6297	0.2646	0.6453
Original J_{AS} (b)	0.9976	0.9986	0.9884	0.9826	0.9995	0.9994	0.9915	0.9797	0.9931	0.9989
Improved J_{AS} (c)	0.9996	0.9978	0.9784	0.9848	0.9995	0.9991	0.9851	0.9976	0.9933	0.9984

TABLE III: F1-scores per class, (a) reference results obtained from [15], (b) original loss function with a single hidden layer network, (c) improved attack-sharing loss function used with a single hidden layer network

	BENIGN	Brute-Force	DDoS	DoS	Infiltration
Original J_{AS} (a)	0.9120	0.5487	0.7960	0.7368	0.3753
Original J_{AS} (b)	0.9981	0.9855	0.9994	0.9856	0.9960
Improved J_{AS} (c)	0.9987	0.9816	0.9993	0.9913	0.9958

IV. Implementation and neural architecture search

Our implementation of the attack-sharing loss relies on the Tensorflow and Scikit-learn Python libraries. In a first step, we prepared the training data by replacing missing values with the corresponding features median value and normalized the input by applying the Scikit-learn StandardScaler. Infinite values were replaced by the maximum value of its feature column. We then split the sample set of about 2.83 million entries with Scikit-learns StratifiedShuffleSplit class into a training set (81 %), a validation set (9 %) and a test set (10 %). The improved attack-sharing loss was implemented as a custom loss function in Tensorflow and it was successfully used in combination with the Nadam [5] optimizer for which we used Tensorflow's default parameters ($\eta = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$). Weight initialization was achieved by the Tensorflow built-in HeNormal algorithm [3]. The batch size was fixed at 512 samples in all our experiments. In order to prevent overfitting, we used a custom early stopping class that was monitoring the validation loss as well as the validation F1-score.

Targeting for the smallest neural network possible in order to deploy it ultimately in an embedded environment, we were experimenting with architectures of different width and depth. We empirically found that a network consisting of a single layer with 110 ReLu activated hidden neurons and 5 Softmax activated output neurons was able to achieve good results. Adding layers however, did not result in a significant increase in performance.

V. Results

To compare our results with previously reported results, we use the F1-score (equation 5) which is less sensitive to imbalanced data than the accuracy measure [6].

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

Recall and precision are defined in terms of the values true positive (TP) which is the number of test samples that were correctly classified, the false positive (FP) which is the number of negative samples that were falsely classified as positive, the false negatives (FN) which are positive cases that were wrongly classified as negatives. The corresponding equations for recall and precision are shown in 6 and 7.

$$Recall = \frac{TP}{TP + FN} \quad (6) \quad Precision = \frac{TP}{TP + FP} \quad (7)$$

To calculate recall and precision for individual classes, equations 6 and 7 are used with TP defined as in equation 8, FP defined by equation 9 and FN defined by equation 10. The term $\hat{y}^{(j)}$ is the j-th prediction, while the variable $y^{(j)}$ is the j-th label. Thus $I(\hat{y}^{(j)}, i)$ equals 1 if the j-th prediction is equal to the i-th class.

$$TP_i = \sum_{j}^{N} I(\hat{y}^{(j)}, i) I(y^{(j)}, i) \tag{8}$$

$$FP_i = \sum_{j}^{N} I(\hat{y}^{(j)}, i)(1 - I(y^{(j)}, i)) \tag{9}$$

$$FN_i = \sum_{j}^{N} (1 - I(\hat{y}^{(j)}, i)) I(y^{(j)}, i) \tag{10}$$

Table II shows the recall and precision values for the individual classes. The first row contains the results for a neural network consisting of 10 layers with 100 neurons each published in [15], the second row shows the values obtained by using the attack-sharing loss for the training of our proposed single layer neural network with just 110 neurons, but with a much longer training of 133 epochs compared to the 10 epochs reported in [15]. The last row of Table II shows the results of the same architecture but trained with our improved attack-sharing loss function. Table III shows the F1-scores for each class. The first row contains the F1-scores calculated from results reported in [15]. The other two rows show our

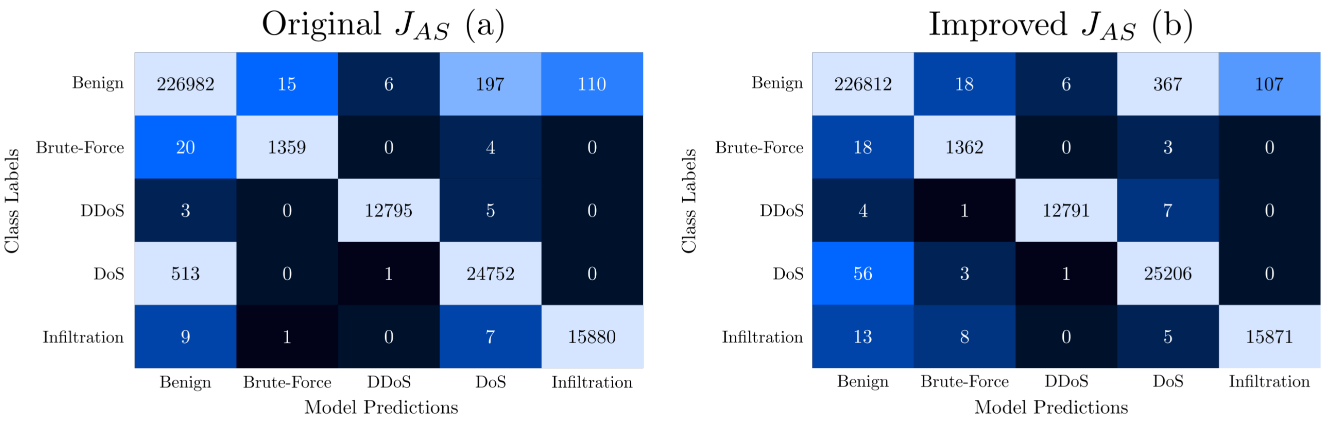

Fig. 1: Confusion matrix attack-sharing loss (a), improved attack-sharing loss (b)

experimental results. Table IV compares the improved attack-sharing loss function and the loss function proposed by Dong et al. [15].

TABLE IV: Comparison of the attack-sharing loss function and improved attack-sharing loss function for a single layer neural network

	F1	Recall	Precision
Original J_{AS}	0.9929	0.9930	0.9940
Improved J_{AS}	0.9933	0.9955	0.9912

The two confusion matrices in Figure 1 give a detailed characterization of the performance obtained with the two loss functions on the same single hidden layer architecture. These results show that the proposed solutions allow to reduce the number of false negatives produced by an artificial neural network (ANN) on the considered network intrusion detection problem.

VI. Discussion

A. Model performance

The comparison of the first two lines of Tables II and III shows that our proposed architecture, in combination with the chosen Nadam optimizer, and a much longer training period, was able to greatly improve the model performance. Comparing the last two rows of the same tables shows that both versions of the attack sharing loss function perform comparably well while using about 9 times fewer resources than the architecture proposed by [15]. Thus, our proposed network used in combination with the improved J_{AS} loss function and well-chosen hyperparameters is better suited to detect harmful activities in a network's flow.

B. Reduction of false negatives

The third row of Tables II and III shows the results of our proposed improved attack-sharing loss function applied to the shallow neural architecture described in Section IV. Comparing these results shows that we were able to achieve a further improvement of the recall value for the attack types Brute-Force and DoS. For the remaining classes, we achieved comparable results. Table IV shows that the improved attack-sharing loss function globally achieved an increased recall and F1-score compared to the original version. Figure 1 shows the confusion matrices for the two loss functions. Every entry of the first column (except the first one) represents an attack that was falsely classified as benign. Ever entry of the first row (except the first one) represents a benign event that was falsely classified as an attack. Comparing the values for the DoS attack shows that we were able to reduce the number false negative classifications from 513 (original J_{AS}) to 56 (improved J_{AS}), which is almost a 10-fold decrease. The corresponding false positive rate was at the same time almost doubled, from 197 (original J_{AS}) to 367 (improved J_{AS}). However, a false positive can be detected downstream, a false negative, on the other hand, passes undetected. Therefore we judge a low false negative rate more important than a low false positive rate.

VII. Conclusions

This paper proposed a shallow neural network capable of successfully detecting computer network attacks with a F1-score above 99%. Shallow networks are computational less demanding and thus better suited for applications in the embedded domain. Furthermore, we proposed an improved version of the attack-sharing loss function. The extension of the parameter λ from the original attack-sharing loss into two separate scaling factors α and β yields the possibility to further tune the training process and hence decrease the false negative rate for certain attack types. It is remarkable that the reported results were obtained with an empirical approach to determine effective hyperparameter values, but it is evident that common hyperparameter optimization methods like for instance grid search are also applicable. The use of a custom loss function to deal with imbalanced multi-class classification problems is quite promising.

VIII. References

[1] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority over-Sampling Technique". In: *J. Artif. Int. Res.* 16.1 (June 2002), pp. 321–357. ISSN: 1076-9757.

[2] Muhammad Atif Tahir, Josef Kittler, and Fei Yan. "Inverse random under sampling for class imbalance problem and its application to multi-label classification". In: *Pattern Recognition* 45.10 (2012), pp. 3738–3750. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.03.014. URL: https://www.sciencedirect.com/science/article/pii/S0031320312001471.

[3] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[4] Zhuoyuan Zheng, Yunpeng Cai, and Ye Li. "Oversampling method for imbalanced classification". In: *Computing and Informatics* 34.5 (2015), pp. 1017–1037.

[5] Timothy Dozat. "Incorporating nesterov momentum into adam". In: (2016).

[6] Guo Haixiang et al. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.12.035. URL: https://www.sciencedirect.com/science/article/pii/S0957417416307175.

[7] Wei Wang et al. "Malware traffic classification using convolutional neural network for representation learning". In: *2017 International Conference on Information Networking (ICOIN)*. 2017, pp. 712–717. DOI: 10.1109/ICOIN.2017.7899588.

[8] Chuanlong Yin et al. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks". In: *IEEE Access* 5 (2017), pp. 21954–21961. DOI: 10.1109/ACCESS.2017.2762418.

[9] Zhida Li, Prerna Batta, and Ljiljana Trajkovic. "Comparison of Machine Learning Algorithms for Detection of Network Intrusions". In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2018, pp. 4248–4253. DOI: 10.1109/SMC.2018.00719.

[10] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:" en. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116. ISBN: 978-989-758-282-0. DOI: 10.5220/0006639801080116. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116 (visited on 04/20/2021).

[11] Haoyang Wang and He Huang. "LAD-SMOTE: A New Oversampling Method Based on Locally Adaptive Distance". In: *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*. 2018, pp. 305–311. DOI: 10.1109/ICICIP.2018.8606712.

[12] Meliboev Azizjon, Alikhanov Jumabek, and Wooseong Kim. "1D CNN based network intrusion detection with normalization on imbalanced data". In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2020, pp. 218–224. DOI: 10.1109/ICAIIC48513.2020.9064976.

[13] Ana Laura Gonzalez Rios et al. "Detection of Denial of Service Attacks in Communication Networks". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180445.

[14] Giuseppina Andresini et al. "GAN augmentation to deal with imbalance in imaging-based intrusion detection". In: *Future Generation Computer Systems* 123 (2021), pp. 108–127. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.04.017. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21001382.

[15] Boxiang Dong et al. *Cyber Intrusion Detection by Using Deep Neural Networks with Attack-sharing Loss*. 2021. arXiv: 2103.09713 [cs.CR].

[16] Lan Liu et al. "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning". In: *IEEE Access* 9 (2021), pp. 7550–7563. DOI: 10.1109/ACCESS.2020.3048198.

[17] Kun Yang, Samory Kpotufe, and Nick Feamster. "An Efficient One-Class SVM for Anomaly Detection in the Internet of Things". In: *CoRR* abs/2104.11146 (2021). arXiv: 2104.11146. URL: https://arxiv.org/abs/2104.11146.

[18] *Intrusion Detection Evaluation Dataset (CICIDS2017)*. https://www.unb.ca/cic/datasets/ids-2017.html. Accessed July 10 2021.