Cyber Intrusion Detection by Using Deep Neural Networks with Attack-sharing Loss

Boxiang Dong*, Hui (Wendy) Wang†, Aparna S. Varde*, Dawei Li*, Bharath K. Samanthula*,
Weifeng Sun‡, Liang Zhao‡
*Montclair State University, Montclair, New Jersey 07043
Email: {dongb, vardea, dawei.li, samanthulab}@montclair.edu
†Stevens Institute of Technology, Hoboken, New Jersey 07030
Email: {Hui.Wang}@stevens.edu
†Dalian University of Technology, Dalian, China 116024
Email: {wfsun, liangzhao}@dlut.edu.cn

***Abstract*—Cyber attacks pose crucial threats to computer system security, and put digital treasuries at excessive risks. This leads to an urgent call for an effective intrusion detection system that can identify the intrusion attacks with high accuracy. It is challenging to classify the intrusion events due to the wide variety of attacks. Furthermore, in a normal network environment, a majority of the connections are initiated by benign behaviors. The class imbalance issue in intrusion detection forces the classifier to be biased toward the majority/benign class, thus leave many attack incidents undetected. Spurred by the success of deep neural networks in computer vision and natural language processing, in this paper, we design a new system named DeepIDEA that takes full advantage of deep learning to enable intrusion detection and classification. To achieve high detection accuracy on imbalanced data, we design a novel attack-sharing loss function that can effectively move the decision boundary towards the attack classes and eliminates the bias towards the majority/benign class. By using this loss function, DeepIDEA respects the fact that the intrusion mis-classification should receive higher penalty than the attack mis-classification. Extensive experimental results on three benchmark datasets demonstrate the high detection accuracy of DeepIDEA. In particular, compared with eight state-of-the-art approaches, DeepIDEA always provides the best *class-balanced accuracy*.**

***Index Terms*—Intrusion detection, Deep learning, Imbalanced classification.**

I. Introduction

Recent years witness an expeditious outbreak of cyber attacks. Online Trust Alliance [1] revealed that 2017 is "the worst year ever" in data breaches and cyber attacks around the world. The amount of disclosed cyber incidents targeting businesses nearly doubled from 82,000 in 2016 to 159,700 in 2017. The penetration attack at Equifax leaked the financial credit report of 145 million consumers, which constitutes 45% of the total population in the U.S. The WannaCry ransomware attack infected 300,000 computer systems within four days, and severely disrupted the medical appointments in the U.K.. These catastrophic attacks bring forth the most intensive aspirations for an effective intrusion detection system (IDS) that can identify the intrusion with high accuracy.

Traditional signature-based IDS techniques heavily depend on the signature database constructed by security experts, and thus fail to detect novel attacks. A wide variety of data mining and machine learning models, e.g., decision tree, support vector machine (SVM), and graph mining algorithms [4], [6], [9], have been adapted to discover anomaly from the network monitoring data. However, they are not favorable at representing intrusion detection classification functions that have many complex variations [5].

Recently, deep learning emerges as a favorable solution to dealing with complicated input-output mappings. Its application in computer vision and natural language processing leads to breakthroughs in these areas. Specifically, it builds a neural network by stacking a certain number of layers of neurons. With sufficiently large number of layers and units, a deep network can represent functions of high complexity. Compared with traditional machine learning models, it avoids the need for feature extraction. Most importantly, it produces the best-in-class accuracy by learning from a large amount of labeled data.

Quite a few latest research in intrusion detection resort to deep learning. Most of them [16], [8] simply learn a new feature representation by using various deep neural networks (e.g., deep autoencoder and convolutional networks), and then rely on traditional classifiers such as SVM and k-nearest neighbor (KNN) to detect attacks. Kitsune [23] is the most recent work that detects network intrusion attacks with deep neural networks. It applies an ensemble of autoencoders to learn the identify function of the original data distribution. For any new instance, its anomaly score is calculated based on the distance between the autoencoders' output and its feature values. However, we argue that such a design only employs deep learning to discover inherent/generic features in network connections, but fails to take advantage of its capacity to learn complex classification functions.

There are two major challenges of designing a deep neural network based intrusion detection system. We discuss these two challenges below.

Challenge 1: diversity of intrusion attacks. There are many types of intrusion attacks, each exploiting a wide range of techniques to conduct the invasion. Even the same type of attacks can exhibit different behavior patterns.

Challenge 2: imbalanced class distribution. In a healthy

978-1-7281-4117-6/19/$31.00 ©2019 IEEE
DOI 10.1109/DataCom.2019.00011

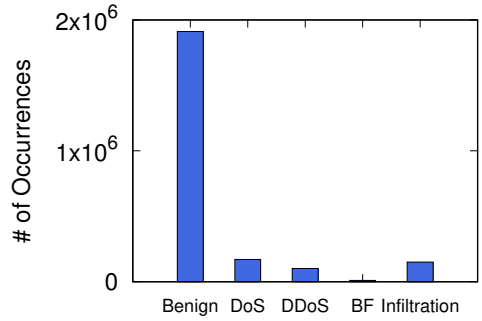

Fig. 1. Class distribution of the CICIDS17 dataset. There are five classes, including the benign event and four types of attacks, namely DoS, DDoS, brute-force (BF), and infiltration attacks.

network environment, a majority of the connections are benign. This makes the network connection instances follow a long-tail class distribution. Moreover, different types of intrusion attacks are unevenly distributed in practice. As an example, consider a real-world network intrusion detection dataset named CICIDS17, which is collected and released by Canadian Institute for Cybersecurity. The data is labeled with 5 classes, including the benign class and four types of attacks, namely DoS, DDoS, brute-force (BF), and infiltration attacks. The distribution of the intrusion attacks is illustrated in Figure 1. Apparently, the classes are highly imbalanced. The imbalanced class distribution forces the classifier models to be biased toward the majority class, and thus lead to poor accuracy on the minority classes (i.e., the intrusions).

To address these two challenges, we build a new intrusion detection and classification framework named DeepIDEA (a Deep Neural Network-based Intrusion Detector with Attack-sharing Loss). DeepIDEA takes full advantage of deep learning to extract features and learn the classification boundary. Besides, we design a new loss function named *attack-sharing loss function* that eliminates the bias towards the majority/benign class by moving the decision boundary towards the attack classes. To our best knowledge, this is the first work on imbalanced deep learning for intrusion detection. Specifically, we make the following contributions.

First, we construct a deep feedforward network to learn intricate patterns of benign communications and malicious connections from the training data. To expedite the learning process on large data, we adapt a novel optimization algorithm that keeps track of an exponentially decaying average of the first-order and second-order moment of past gradients to dynamically adjust the learning rate.

Second, to address the class imbalance problem in intrusion detection, we design a new loss function named attack-sharing loss for our deep feedforward network. The attack-sharing loss function takes the discrepancy penalty of different types of mis-classification (e.g., mis-classifying attack types versus mis-classifying intrusion as normal) into consideration, so that the mis-classification of intrusions as benign receives more penalty than the mis-classification of attacks. It can be integrated with any deep neural network to mitigate the bias towards the majority class.

Last but not least, we launch an extensive set of experiments on three benchmark datasets. The comparison with 8 baseline approaches demonstrate the effectiveness of DeepIDEA. In particular, DeepIDEA produces the best detection accuracy on every dataset.

The rest of the paper is organized as follows. Section II discusses the background information. Section III presents our the design of DeepIDEA. Section IV shows the experiment results. Section V introduces the related work. Finally, Section VI concludes the paper.

II. Background

In this section, we introduce the background knowledge, including the concepts of deep neural network, intrusion attacks, and imbalanced classification.

A. Deep Neural Network

Multi-layer perceptrons (MLPs), also known as deep feed-forward network, is a network that consists of an input layer, multiple hidden layers, and an output layer. Each layer includes a certain number of neurons/units. The neurons in consecutive layers are connected by links with certain weights. Besides MLPs, other specialized architectures have been proposed in recent years. For example, convolutional networks are known for image processing, while recurrent neural networks are specialized at capturing long term dependencies [13]. Learning the parameters (i.e., weights and bias) of a neural network is typically solved by using gradient descent. Back propagation provides an efficient way to calculate the gradients so as to optimize the weights associated with the connections. Various optimization algorithms were proposed to acclerate the learning process, e.g., stochastic gradient descent (SGD) [21], Nesterov Momentum [22], and Adam optimizer [18].

B. Intrusion Attacks

Numerous types of network intrusion attacks make it intricate to design an effective detection method. Next, we briefly introduce five prevailing attacks that are investigated in this paper.

- *Brute-Force* attack is the most simple attack to gain illegal access to a site or server. The most common brute-force attack is the dictionary attack that cracks user passwords. There have been a few successful brute-force attacks. For example, in 2016, a massive brute-force attack against Alibaba Inc, a Chinese e-commerce giant, compromised 20.6 million accounts [2].
- *Botnet* attack exploits a number of Internet-connected devices (zombies) to carry out malicious and criminal tasks. As an example, a recent study [12] of 6 million Twitter accounts reveals that 350,000 of them are zombies of the Star Wars botnet.
- *Probing* attack scans a victim device in order to determine the vulnerabilities that can be exploited to compromise the system. It usually uses a network mapper (e.g.,

Nmap[1]) to send TCP packets to discover vulnerable hosts and services.

- *DoS/DDoS* attack overloads the target machine and prevents it from serving the intended users. DDoS is different from DoS mainly in that it leverages multiple systems to exhaust the resources of the victim. The famous 2016 DDoS attack against Dyn DNS [3] made it impossible for users to connect to Amazon, GitHub, Spotify, etc.
- *Infiltration* attack leverages the vulnerability in particular software such as Adobe Acrobat Reader to execute a backdoor on the target computer. Once the attack is successful, the attacker can launch various types of attacks against the victim's network, including IP sweep and port scan.

C. Imbalanced Classification

In general, the intrusion detection problem can be modeled as a classification problem in machine learning, by which the classification model outputs if the network system is intruded or not. In most real-world datasets, the data labels follow a long tail distribution, i.e., some specific classes are represented by a very small number of instances compared to other classes. In the scenario of intrusion detection, the data is also imbalanced, as the benign network behaviors dominate the collected dataset, while the intrusion events are rarely observed. To improve the overall accuracy, the imbalanced data forces the classification model to be biased toward the majority classes. This class imbalance problem renders poor accuracy on detecting intrusion attacks, as intrusion classes are under-represented. Several approaches [27] try to mitigate the negative effects of imbalanced data for general classification problems. One solution is to do *over-sampling* [15] or *under-sampling* [20]. In particular, over-sampling on the under-represented classes duplicates these instances, while *under-sampling* [20] eliminates samples in the over-sized classes. However, over-sampling often leads to over-fitting and longer training time, and under-sampling degrades the overall accuracy since it discards potentially useful training instances. Another solution is to make the loss function cost-sensitive by associating larger error penalty with under-represented classes [17]. However, in deep learning, such cost-sensitive loss function can make the loss of a minibatch highly sensitive to the label distribution. As the consequence, it leads to non-convergence of the training process, and potentially inferiors the decision boundary of the classifier.

III. Our Approach

In this section, we present the details of our intrusion detector, namely DeepIDEA. In particular, we will discuss the model, loss function, and the optimization procedure of DeepIDEA.

A. Model

At high level, DeepIDEA is built as a fully-connected neural network with one input layer, $L > 1$ hidden layers, and one output layer. The input layer consists of d units, each representing an input feature. The architecture of DeepIDEA is illustrated in Figure 2. Next, we explain the details of the model.

We denote each input data point as $(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is the set of features, and y is the label. The input layer of DeepIDEA consists of d neurons that take input features. For the sake of simplicity, for any instance $(\boldsymbol{x}, y)$, we assume $\boldsymbol{h}^{(0)} = \boldsymbol{x}$. We use $\boldsymbol{h}^{(\ell)}$ to denote the output of the ℓ-th hidden layer ($1 \leq \ell \leq L$). Let $n_i^{(\ell)}$ be the i-th unit of the ℓ-th hidden layer. The output of $n_i^{(\ell)}$ is computed as:

$$h_i^{(\ell)} = g(\boldsymbol{w}_i^{(\ell)} \tilde{\boldsymbol{h}}^{(\ell-1)} + b_i^{(\ell)}), \tag{1}$$

where $g(x) = max\{0, x\}$ is the *rectified linear units* (ReLU) activation function, $\boldsymbol{w}_i^{(\ell)}$ is the i-th row in the weight matrix that connects the $(\ell-1)$-th and ℓ-th layer, $b^{(\ell)}$ is the bias vector at the ℓ-th layer, and $\tilde{\boldsymbol{h}}^{(\ell-1)}$ is the thinned output from the $(i-1)$-th layer by using dropout. In particular,

$$\tilde{\boldsymbol{h}}^{(\ell-1)} = \boldsymbol{h}^{(\ell-1)} * \boldsymbol{r}, \tag{2}$$

where $*$ is the Hadamard product operator, $\boldsymbol{r}$ is a mask vector that specifies which units to be used. In particular, $\boldsymbol{r}$ consists of independent Bernoulli random variables, each of which has probability p of being 1, i.e., $r_i \sim Bernoulli(p)$.

The output layer of the neural network includes c *softmax* units, where c is the number of classes. For any instance $(\boldsymbol{x}^{(i)}, y^{(i)})$, the predicted probability that it belongs to the j-th class $p_j^{(i)}$ is computed as

$$p_j^{(i)} = softmax(\boldsymbol{z})_j = \frac{exp(z_j)}{\sum_{k=1}^{c} exp(z_k)}, \tag{3}$$

where $\boldsymbol{z}$ is a vector of linear activations of the output layer.

B. Attack-sharing Loss Function

Most modern neural networks use *cross-entropy loss* J_{CE} to describe the discrepancy between the ground-truth labels and the model predictions. In particular, the loss J_{CE} is calculated as:

$$\begin{aligned} J_{CE}(\boldsymbol{\theta}) &= \mathbb{E}_{(\boldsymbol{x}^{(i)}, y^{(i)}) \sim \hat{p}_{data}} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) \\ &= -\mathbb{E}_{(\boldsymbol{x}^{(i)}, y^{(i)}) \sim \hat{p}_{data}} \log p(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \\ &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{c} \mathbf{I}(y^{(i)}, j) \log p_j^{(i)}, \end{aligned} \tag{4}$$

where $\boldsymbol{\theta}$ consists of the weight matrix between consecutive layers in the neural network, $\hat{p}_{data}$ is the empirical data distribution in the training set, $p(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ is the probability that the neural network correctly classifies the input $\boldsymbol{x}^{(i)}$, N is the number of training samples, c is the number of classes, and $\mathbf{I}$ is the indicator function s.t.

$$\mathbf{I}(a, b) = \begin{cases} 1 & \text{if a=b} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The parameters $\boldsymbol{\theta}$ in the network are optimized so as to minimize $J_{CE}(\boldsymbol{\theta})$ and obtain the desired classification accuracy.

[1] https://nmap.org/

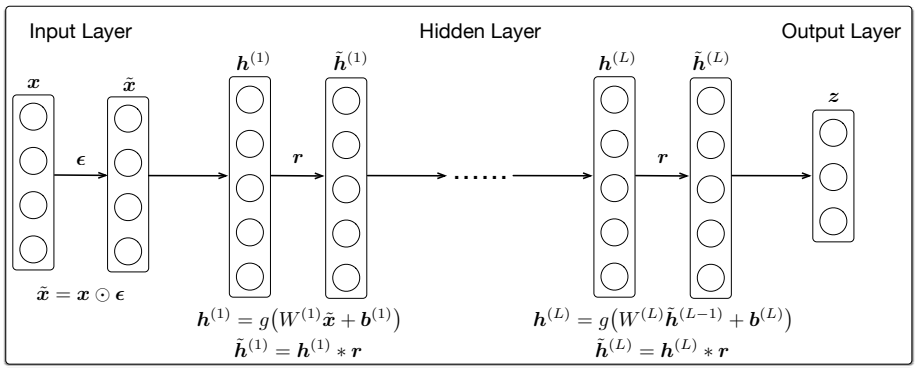

Fig. 2. The model structure of DeepIDEA (Tildes indicate regularized layers)

One weakness of the cross-entropy loss function is that it does not take the type of mis-classification into consideration, and thus penalizes the classification error for all classes equally. There are two types of mis-classification for the intrusion detection system:

- *Intrusion mis-classification*: an intrusion attack is mis-classified as benign event; and
- *Attack mis-classification*: an intrustion attack of type A (e.g., DoS attack) is mis-classified as an intrution attack of type B (e.g., probing attack).

In practice, the intrusion mis-classification should be penalized more than the attack mis-classification, as the attack mis-classification still triggers an alert to the IT security team, and enables the incident to be further inspected, whereas the intrusion mis-classification enables the attack incidents to bypass the security check and cause potentially critical damage. Therefore, the intrusion mis-classification should have higher penalty than the attack mis-classification.

To address the issue of discrepancy penalty of different types of mis-classification, we improve the basic cross-entropy loss function. In particular, for any instance $(\boldsymbol{x}^{(i)}, y^{(i)})$, if it is a benign incident, $y^{(i)} = 1$; otherwise, $y^{(i)} \in \{2, \ldots, c\}$. Inspired by [24], we design the *attack-sharing loss function*, J_{AS}, with an additional regularization term that penalizes intrusion mis-classification, i.e., the wrong estimation between the benign label and attack labels. In particular, we have

$$J_{AS} = J_{CE} - \frac{1}{N}\sum_{i=1}^{N} \lambda\big(\mathbf{I}(y^{(i)}, 1)\log p_1^{(i)} + \sum_{j=2}^{c} \mathbf{I}(y^{(i)}, j)\log(1 - p_1^{(i)})\big),] \tag{6}$$

where λ is a control parameter. When λ is small, J_{AS} is similar to the vanilla cross-entropy loss unction; when λ is large, J_{AS} tends to be an objective function for addressing the binary classification problem, benign versus attack. In our experiments, we set $\lambda = 10$. Compared with the basic cross-entropy loss, the attack-sharing loss function eliminates the bias towards the majority/benign class by moving the decision boundary towards the attack classes. It also respects the discrepancy penalty of different types of mis-classification.

C. Optimization Procedure

In deep learning, the most widely-used optimization algorithm is *stochastic gradient descent* (SGD). In each round, it uses a minibatch of samples to estimate the gradient, and updates the parameters. Although simple, SGD suffers from slow asymptotic convergence, especially when there exist saddle points (i.e. points where one dimension slopes up and another slopes down) and plateaus (i.e., areas where the gradients keep stably high) in the parameter space. Due to the complicated nature of intrusion detection classification boundary, saddle points and plateaus widely exist. To expedite the learning process, we adapt the *Adam* optimizer, which adaptively updates the learning rate. In particular, we employ two variables, $\boldsymbol{s}$ and $\boldsymbol{r}$, to store an exponentially decaying average of past gradients and squared gradients respectively. Initially, we set $\boldsymbol{s} = \mathbf{0}$ and $\boldsymbol{r} = \mathbf{0}$. In the t-th round of feedforward and backpropagation, we take a minibatch of m samples from the training set, and calculate the stochastic gradient:

$$\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}_{t-1}} J(\boldsymbol{\theta}_{t-1}). \tag{7}$$

Next, we update the first-order moment and second-order moment:

$$\boldsymbol{s}_t = \rho_1 \boldsymbol{s}_{t-1} + (1 - \rho_1)\boldsymbol{g}_t, \tag{8}$$

$$\boldsymbol{r}_t = \rho_2 \boldsymbol{r}_{t-1} + (1 - \rho_2)\boldsymbol{g}_t^2, \tag{9}$$

where $\rho_1, \rho_2 \in (0, 1)$ are the hyperparameters that determine the decay rate. We also perform bias corrections to both moments to account for their initialization at the origin:

$$\boldsymbol{s}_t = \frac{\boldsymbol{s}_t}{1 - \rho_1^t}, \tag{10}$$

$$\boldsymbol{r}_t = \frac{\boldsymbol{r}_t}{1 - \rho_2^t}. \tag{11}$$

Finally, the parameters are updated as

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\zeta \boldsymbol{s}_t}{\sqrt{\boldsymbol{r}_t} + \delta}, \tag{12}$$

where ζ is the step size, and δ is a small stabilization factor. By using Equation (12), we make greater evolution in the more gently sloped directions of parameter space. This facilitates faster convergence compared with SGD. Another attractive property of the *Adam* optimizer is that it is robust to the choice of hyperparameters.

IV. Experiments

A. Dataset

In our experiments, we use three datasets, namely *KDD99*, *CICIDS17* and *CICIDS18* dataset. Next, we briefly introduce these datasets.

TABLE I
CLASS DISTRIBUTION IN KDD99 DATASET

Label	Training		Testing	
	Number	Fraction	Number	Fraction
Benign	972,781	19.86%	60,593	19.48%
DoS	3,883,390	79.28%	231,455	74.42%
Probing	41,102	0.84%	4,166	1.34%
U2R	52	0.01%	245	0.08%
R2L	1,106	0.02%	14,570	4.68%
Total	4,898,431	100%	311,029	100%

KDD99 dataset is built by Stolfo et al. [25], as a part of the DARPA Intrusion Detection Evaluation Program. To prepare this dataset, a military network environment was deployed to acquire nine weeks of raw TCP dump data from a local-area network (LAN). The LAN was operated as if it were a typical U.S. Air Force LAN, except that it was hacked by a sequence of cyber attacks. In this dataset, each connection record is described by 41 features and 1 label. The features include information in three aspects, namely basic connection information (e.g., duration, protocol type (tcp, udp, icmp), number of wrong fragments, number of urgent packets, etc), content information (e.g., number of failed login attempts, number of shell prompts, number of operations on access control files, etc), and traffic information (e.g., number of connections to the host in the past two seconds, fraction of connections that have "SYN" errors, etc). The attacks in the dataset fall into 4 categories, i.e., DoS, Probing, U2R (normal users illegally gain root access to the system), and R2L (remote attackers exploit some vulnerabilities to obtain local access to the host). We show the distribution of these attacks in the training and testing set in Table I.

TABLE II
CLASS DISTRIBUTION IN CICIDS17 DATASET

Label	Training		Testing	
	Number	Fraction	Number	Fraction
Benign	1,911,674	81.57%	361,399	74.84%
DoS	170,508	7.27%	82,151	17.01%
DDoS	101,024	4.31%	27,003	5.59%
Brute-Force	10,494	0.45%	3,341	0.69%
Infiltration	149,934	6.40%	9,032	1.87%
Total	2,343,634	100%	482,926	100%

CICIDS17 dataset is collected by Canadian Institute for Cybersecurity and is publicly available[2]. Two networks, namely the attack network and victim network were constructed. Each network is a infrastructure that consists of routers, switches and a set of PCs running most of the common operating systems. In total, there are 2.83 million network connection instances, where each instance is described by 81 features. The features are similar to those of the KDD99 dataset dataset, but include more statistical information. We omit the details due to the space limit. We manually split the dataset into a training set and a testing set with a $5:1$ size ratio. The launched attacks include DoS, DDoS, Infiltration and Brute-Force attacks. We show the distribution of these attacks in the training and testing set in Table II.

TABLE III
CLASS DISTRIBUTION IN CICIDS18 DATASET

Label	Training		Testing	
	Number	Fraction	Number	Fraction
Benign	4,197,451	82.62%	814,704	76.62%
DoS	517,691	10.19%	158,098	14.87%
Infilteration	131,844	2.60%	38,787	3.65%
Botnet	233,085	4.59%	51,753	4.87%
Total	5,080,071	100%	1,063,342	100%

CICIDS18 dataset is also constructed by Canadian Institute for Cybersecurity and is publicly available[3]. To simulate a real-world network, a common LAN network topology is implemented on the AWS computing platform. The attacking infrastructure includes 50 machines and the victim organization has 5 departments and includes 420 machines and 30 servers. The dataset includes 6.3 million network connection instances, and each instance has 77 features. Again, we manually split the dataset into a training set and a testing set with a $5:1$ size ratio. The attacks in this dataset includes DoS, Infiltration and Botnet. The class distribution of these attacks can be found in Table III.

TABLE IV
SUMMARY OF THE DATASETS

Dataset	# of Features	Training Size	Testing Size	# of Classes	Ω_{imb}
KDD99	41	4,898,431	311,029	5	2.96
CICIDS17	81	2,343,634	482,926	5	3.08
CICIDS18	77	5,080,071	1,063,342	4	2.31

We summarize the characteristics of these three datasets in Table IV. To evaluate the level of class imbalance in each dataset, we also report the *class imbalance measure* Ω_{imb} [10] in the training set, which is defined as

$$\Omega_{imb} = \frac{\sum_{i=1}^{c} n_{max} - n_i}{n}, \tag{13}$$

where n denotes the number of instances in the datast, n_i denotes the number of instances that belong to the i-th class, and $n_{max} = max_{i=1}^{c} n_i$. Intuitively, Ω_{imb} measures the minimum percentage count of data samples required over all classes in order to form an overall balanced/uniform distribution. A larger Ω_{imb} value indicates higher level of class imbalance.

B. Setup

We implement DeepIDEA in Python by using the *tensorflow* framework. In our neural network, we include 100 units in each hidden layer. We try various number of hidden layers. Our results suggest that the performance is reasonably good as long as the network consists of at least 6 layers. In the following section, we only show the results with 10 hidden layers. The keep probability of each dropout layer is 0.8. The source code is available in a public repository[4]. In our experiments, we group 100 consecutive network connection instances as a training sample. We set the learning rate as

[2]https://www.unb.ca/cic/datasets/ids-2017.html

[3]https://www.unb.ca/cic/datasets/ids-2018.html

[4]https://github.com/bxdong7/MLP-D

TABLE V
DETECTION ACCURACY COMPARISON BETWEEN DEEPIDEA AND THE BASELINES ON THE KDD99 DATASET

Classifier	Benign		DoS		Probe		U2R		R2L		CBA
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	
SVM	30.2	70.04	96.66	98.55	13.68	27.71	21.87	6.76	84.4	5.91	41.23
KNN	23.51	55.16	90.97	100	100	26.91	0	0	0	0	16.41
DT	20.64	48.23	88.9	99.87	0	0	0	0	0	0	29.49
MLP+CE	0	0	70.92	100	5.36	15.33	0	0	72.29	6.4	24.27
MLP+OverSampling [15]	27.15	11.68	95.5	83.14	5.64	74.64	11.83	1.97	62.79	19.15	38.12
MLP+UnderSampling [20]	19.21	45.95	95.52	83.57	9.43	33.87	20.83	8.09	68.3	29.78	40.25
Cost-Sensitive [17]	33.15	10.59	0	0	4.82	8.27	2.55	73.76	0	0	18.26
CNN [8]	20.50	64.79	94.88	82.81	10.53	27.27	7.14	1.3	60.87	4.67	36.17
DeepIDEA(Our approach)	19.46	85.82	93.03	85.68	7.14	47.87	0	0	62.09	8.89	**45.31**

TABLE VI
DETECTION ACCURACY COMPARISON BETWEEN DEEPIDEA AND THE BASELINES ON THE CICIDS17 DATASET

Classifier	Benign		DoS		DDoS		Brute-Force		Infiltration		CBA
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	
SVM	86.42	76.38	96.58	53.74	92.62	16.03	0	0	7.27	86.18	46.47
KNN	91.92	85.05	75.88	48.22	72.56	86.23	0	0	10.92	84.75	60.85
DT	66.51	100	0	0	0	0	0	0	0	0	20
MLP+CE	87.04	90.76	74.12	63.69	74.73	79.53	7.37	4.8	28.03	61.54	60.06
MLP+OverSampling [15]	86.03	95.05	80.14	52.5	56.68	76.06	3.65	1.63	28.18	53.62	55.45
MLP+UnderSampling [20]	86.88	54.9	50.91	59.31	26.13	11.32	7.17	27.39	13.8	58.03	42.19
Cost-Sensitive [17]	61.58	61.17	17.69	28.09	0	0	0	0	0	0	17.85
CNN [8]	0	0	23.42	96.04	0	0	8.07	11.07	0	0	21.42
DeepIDEA(Our approach)	88.5	94.06	88.77	62.97	76.31	83.19	8.29	4.1	26.46	64.53	**61.77**

TABLE VII
DETECTION ACCURACY COMPARISON BETWEEN DEEPIDEA AND THE BASELINES ON THE CICIDS18 DATASET

Classifier	Benign		DoS		Infiltration		Botnet		CBA
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	
SVM	76.59	42.59	44.97	26.23	26.57	20.64	13.70	29.14	29.65
KNN	75.46	30.05	31.2	28.05	11.53	6.45	17.32	28.14	23.18
DT	64.77	100.0	0	0	0	0	0	0	25.00
MLP+CE	67.42	71.73	31.92	45.23	0	0	18.52	0.42	29.35
Cost-Sensitive [17]	54.52	19.6	0	0	5.44	71.42	10.38	6.54	24.39
CNN [8]	0	0	22.44	98.52	7.08	8.19	0	0	26.68
DeepIDEA(Our approach)	70.96	39.7	45.47	49.94	6.88	21.0	11.86	33.64	**36.07**

10^{-4}, the minibatch size as 128, and the number of epoches as 10. We use cross-validation to avoid overfitting. We run DeepIDEA on a workstation with 1 NVIDIA RTX 2080 Ti GPU and 1 Intel i7-7700K @ 4.2GHz CPU. On average, the training process halts within 3 hours.

C. Baseline

We compare DeepIDEA with the following baseline approaches.

- SVM (support vector machine) with 100 iterations
- KNN (k-nearest neighbor) with 5 neighbors and minkowski distance
- DT (decision tree) with 10 layers at most
- MLP+CE (multi-layer perceptron with cross-entropy loss function (Formula (4)))
- MLP+OverSampling [15]: In this approach, the training data is transformed to balanced class distribution by applying over-sampling.
- MLP+UnderSampling [20]: In this approach, the training data is transformed to balanced class distribution by applying over-sampling.
- Cost-Sensitive [17]: This approach uses cost-sensitve loss function to train the neural network. We follow [17] to set up the cost matrix.
- CNN [8]: This approach classifies the network connections with 2 convolution layers, 2 maxpooling layers and 6 fully-connected layers.

We implement these machine learning-based baselines by using the *scikit-learn* library, and the deep learning-based baselines by using the *tensorflow* framework. We also implement a baseline approach based on recurrent neural network with LSTM units [14]. However, the training process does not complete within 5 days. Therefore, we do not include the results in this section.

D. Evaluation Metrics

In this paper, we are mostly interested in evaluating the detection accuracy. In the experiments, we collect the following statistics: TP (true positive), FP (false positive), TN (true negative) and FN (false negative). We measure the following the *precision* and *recall* for each class. In particular, $precision = \frac{TP}{TP+FP}$ measures the fraction of alerts indicated by the detection approach that are correct, and $recall = \frac{TP}{TP+FN}$ measures the fraction of attacks that are successfully identified by the detection approach.

In addition, we measure the overall class-balanced accuracy (CBA) [10] for all classes. CBA is calculated as the average recall for all classes. By taking all classes equally important, it

avoids inflated performance estimates on imbalanced datasets. If the classifier performs equally well on every class, this term is equivalent to the conventional accuracy (i.e., the number of correct predictions divided by the total number of predictions). On the contrary, if the model only performs well on the majority class, CBA drops to $\frac{1}{c}$. Therefore, CBA is an effective metric in measuring the accuracy of a classifier for imbalanced dataset.

E. Evaluation of DeepIDEA

KDD99 Dataset. We present the classification accuracy of DeepIDEA and the baseline approaches on the KDD99 dataset in Table V. First, we observe that DeepIDEA yields the highest CBA. This demonstrates that DeepIDEA is effective in detecting intrusion attack incidents from imbalanced dataset. Second, KNN, DT and MLP+CE only focus on the majority classes, and produces unsatisfactory performance on the minority classes. For instance, KNN and DT fail to catch any U2R and R2L attack instance. In contrast, the cost-sensitive classifier associates too much cost for the U2R and R2L classes, as they are extremely under-represented. This makes the classifier lean too much toward these classes, and performs poorly on the KDD99 dataset. Over-sampling and under-sampling mitigate the side-effect of the class imbalance problem, but the improvement is not comparable with DeepIDEA.

CICIDS17 Dataset. We report the accuracy of all the classifiers in Table VI. First, we observe that DeepIDEA produces similar and satisfying precision and recall on every class, except for the *Brute-Force*. Consequently, DeepIDEA yields the best CBA among all the classifiers. In Table II, it is easy to see that *Brute-Force* is the most under-represented class. The *Brute-Force* attack instances only takes around 0.5% of the dataset. Even though the attack-sharing loss function aims at dragging the decision boundary toward the attack classes, it does not help much with this class. Second, the performance of KNN and MLP+CE are close to that of DeepIDEA. We further investigate the reason and find that a large fraction of the attack instances are present in both the training and testing set. DT simply labels every test instance as benign connection. Similarly, CNN [8] almost recognizes every connection as DoS attack. The cost-sensitive classifier only focuses on the benign and DoS class. Naturally, the CBA of these three baselines are low.

CICIDS18 Dataset. In Table VII, we compare the accuracy of DeepIDEA with the baselines on the CICIDS18 dataset. DeepIDEA performs the second best in addressing the class imbalance problem. It has similar recall on every class. Again, it shows the effectiveness of the class-sharing loss function. No baseline approach produces a CBA higher than 30%. Again, the cost-sensitive baseline concentrates all the attention to the most under-represented class, i.e., infilteration, and neglects the other classes. DT simply recognizes every testing instance as benign.

F. Insights

On all datasets, DeepIDEA delivers the best CBA. The CBA of DeepIDEA is always better and can as twice high as that of MLP+CE. Compared with the sampling-based approaches, the accuracy of DeepIDEA on all classes is more balanced. DeepIDEA significantly outperforms the cost-sensitive classifier [17], which only focuses on a few classes. CNN [8] is inferior to DeepIDEA in classification accuracy mainly because network events that are close in time do not exhibit similar behaviors. All these observations demonstrate the effectiveness of our attack-sharing loss function in dealing with class imbalance in intrusion detection dataset.

However, we must acknowledge the weakness of DeepIDEA. It does not concentrate sufficiently on the extremely under-represented classes. For example, the U2R class in the KDD99 dataset and the Brute-Force class in the CICIDS17 dataset are rarely detected by DeepIDEA. Both classes take no more than 1% in the training and testing set. The reason is that the attack-sharing loss only pulls the decision boundary towards the attack classes. However, the tow direction is still biased towards the majority attack classes. In specific, the regularization term in Formula (6) does not differentiate different types of attacks. In order to minimize the J_{AS} loss, the classifier tends to classify every attack instance as a majority attack class. This limitation makes DeepIDEA more suitable for the scenarios where the benign instances dominate the dataset, and different types of attacks are balanced. In other words, DeepIDEA works well when the imbalance only exists between the benign class and attack classes.

V. Related Work

A. Intrusion Detection based on Deep Learning

[4], [6] conducted a thorough comparison of a variety of classifiers, including J48 tree, Naive Bayes, decision tree, support vector machine (SVM) and k-nearest neighbor (KNN), on the accuracy of intrusion detection. To reduce the generalization error of the classifiers and avoid overfitting, Rigaki et al. Javaid et al. [16] developed a novel intrusion detection approach based on *self-taught learning*. It is a deep neural network that consists of two stages. In the first stage, a new feature representation of the input data is learned from a sparse auto-encoder. After that, the new features are taken by a soft-max regression for classification. The experiment results demonstrate the effectiveness of the new feature representation in improving classification accuracy. Chowdhury et al. [8] applied 'few-shot learning' to improve the detection accuracy. In specific, they first train a convolutional neural network and then extract features from various layers. Those features are fed into various classifiers such as SVM or 1-nearest neighbor (1-NN). Experimental results suggest that the class-wise accuracy can reach 65.8%. However, it is not clear what is the rationale behind the choice of SVM and 1-NN over a softmax activation in the last layer of a deep neural network. Kitsune [23] is the most recent work that detects network intrusion attacks with deep neural networks. It applies

an ensemble of autoencoders to learn the identify function of the original data distribution. For any new instance, its anomaly score is calculated based on the distance between the autoencoders' output and its feature values. However, we argue that such a design only employs deep learning to discover inherent/generic features in network connections, but fails to take advantage of its capacity to learn complex classification functions.

B. Anomaly Detection based on Deep Learning

Quite a few work concentrate on applying deep neural networks to detect abnormal behaviors that are not essentially intrusion attacks. Zhang et al. [26] propose to detect IT system failures from system log files. Clustering analysis is used to extract frequent patterns in the log files. Then the log files are embedded by counting the number of occurrences of these patterns. To cope with the lack of labeled data, a LSTM is trained to capture the long-range dependency across sequences. Du et al. [11] also applied LSTM to detect anomalies and diagnoze failures from system logs. Different from previous work, they aim at abnormal execution paths to improve the confidence and help with further investigation. Kiran et al. [19] provide a comprehensive survey on the application of deep learning over anomaly detection in videos. Recently, Chalapathy et al. [7] show that it is advantageous to directly train deep networks to extract progressively rich representation of data with the classification objective, rather than utilizing a hybrid approach where deep features are firstly learned by using an autoencoder and then fed into a separate anomaly detection method like SVM. Our work is consistent with this proposition, and thus produces higher accuracy compared with traditional hybrid approaches.

VI. Conclusion

In this paper, we design DeepIDEA, a novel intrusion detection and classification system based on imbalanced deep learning. To deal with the imbalanced class problem, we design a new loss function named attack-sharing loss to eliminate the bias towards the majority/benign class. We also integrate DeepIDEA with a new optimization algorithm to facilitate efficient training. Experimental results on three benchmark datasets show the superiority of DeepIDEA compared to seven baseline approaches.

In the future, we plan to extend this work from the following perspectives. First, we plan to apply the attack-sharing loss to recurrent neural networks that take the network context into consideration. We also plan to use generative adversarial networks to improve the detection accuracy.

References

[1] Alibaba security fail: Brute-force bonanza yields 21m logins. https://www.theregister.co.uk/2016/02/08/alibaba_taobao_security_process_failure/

[2] Cyber incident & breach trends report. https://otalliance.org/system/files/files/initiative/documents/ota_cyber_incident_trends_report_jan2018.pdf

[3] Dyn analysis summary of friday october 21 attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[4] Almseidin, M., Alzubi, M., Kovacs, S., Alkasassbeh, M.: Evaluation of machine learning algorithms for intrusion detection system. In: IEEE International Symposium on Intelligent Systems and Informatics (SISY). pp. 277–282 (2017)

[5] Bengio, Y., et al.: Learning deep architectures for ai. Foundations and trends® in Machine Learning **2**(1), 1–127 (2009)

[6] Biswas, S.K.: Intrusion detection using machine learning: A comparison study (2018)

[7] Chalapathy, R., Menon, A.K., Chawla, S.: Anomaly detection using one-class neural networks. arXiv preprint arXiv:1802.06360 (2018)

[8] Chowdhury, M.M.U., Hammond, F., Konowicz, G., Xin, C., Wu, H., Li, J.: A few-shot deep learning approach for improved intrusion detection. In: IEEE Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON). pp. 456–462 (2017)

[9] Dong, B., Chen, Z., Wang, W.H., Tang, L.A., Zhang, K., Lin, Y., Li, Z., Chen, H.: Efficient discovery of abnormal event sequences in enterprise security systems. In: Proceedings of the ACM International Conference on Conference on Information and Knowledge Management (CIKM) (2017)

[10] Dong, Q., Gong, S., Zhu, X.: Imbalanced deep learning by minority class incremental rectification. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)

[11] Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298 (2017)

[12] Echeverria, J., Zhou, S.: The star wars botnet with¿ 350k twitter bots. arXiv preprint arXiv:1701.02405 (2017)

[13] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

[14] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[15] Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent Data Analysis **6**(5), 429–449 (2002)

[16] Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). pp. 21–26 (2016)

[17] Khan, S.H., Hayat, M., Bennamoun, M., Sohel, F.A., Togneri, R.: Cost-sensitive learning of deep feature representations from imbalanced data. IEEE transactions on neural networks and learning systems **29**(8), 3573–3587 (2018)

[18] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[19] Kiran, B.R., Thomas, D.M., Parakkal, R.: An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. Journal of Imaging **4**(2), 36 (2018)

[20] Kubat, M., Matwin, S., et al.: Addressing the curse of imbalanced training sets: one-sided selection. In: Icml. vol. 97, pp. 179–186. Nashville, USA (1997)

[21] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[22] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119 (2013)

[23] Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089 (2018)

[24] Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3982–3991 (2015)

[25] Stolfo, S.J., Fan, W., Lee, W., Prodromidis, A., Chan, P.K.: Cost-based modeling for fraud and intrusion detection: Results from the jam project. Tech. rep., Columbia University (2000)

[26] Zhang, K., Xu, J., Min, M.R., Jiang, G., Pelechrinis, K., Zhang, H.: Automated it system failure prediction: A deep learning approach. In: IEEE International Conference on Big Data. pp. 1291–1300 (2016)

[27] Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge & Data Engineering (1), 63–77 (2006)