IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2019, accepted August 1, 2019, date of publication August 5, 2019, date of current version August 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933304

Learning Multilevel Auto-Encoders for DDoS Attack Detection in Smart Grid Network

SHAN ALI AND YUANCHENG LI
School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China
Corresponding author: Yuancheng Li (ycli@ncepu.edu.cn)

This work was supported in part by the State Grid Corporation Headquarters Science and Technology Project "Research and Application of Key Technologies for Open Source Software Security Monitoring" under Grant SGFJXT00YJJS1800074.

ABSTRACT Bidirectional communication infrastructure of smart systems, such as smart grids, are vulnerable to network attacks like distributed denial of services (DDoS) and can be a major concern in the present competitive market. In DDoS attack, multiple compromised nodes in a communication network flood connection requests, bogus data packets or incoming messages to targets like database servers, resulting in denial of services for legitimate users. Recently, machine learning based techniques have been explored by researchers to secure the network from DDoS attacks. Under different attack scenarios on a system, measurements can be observed either in an online manner or batch mode and can be used to build predictive learning systems. In this work, we propose an efficient DDoS attack detection technique based on multilevel auto-encoder based feature learning. We learn multiple levels of shallow and deep auto-encoders in an unsupervised manner which are then used to encode the training and test data for feature generation. A final unified detection model is then learned by combining the multilevel features using and efficient multiple kernel learning (MKL) algorithm. We perform experiments on two benchmark DDoS attack databases and their subsets and compare the results with six recent methods. Results show that the proposed method outperforms the compared methods in terms of prediction accuracy.

INDEX TERMS Auto-encoder, cyber security, DDoS attack detection, multiple kernel learning, smart grid.

I. INTRODUCTION

Smart grid is an electrical supply network that combines an existing power network with modern information technologies to respond more efficiently to the needs and distribution of energy. It offers several novel features that include bi-directional communication, remote controlling of smart home appliances, updates about consumer behavior and keeping track of power grid's stability. Such novel features need integration of new services and devices as well as new standards and protocols for effective and simplified operation. However, the incorporation of all these standards and devices increases the complexity and vulnerability of the smart grid to security threats. Particularly, the bidirectional and software-oriented nature of the smart grid makes it very prone to cyber attacks. A cyber attack can have a significant impact on the whole grid that eventually will affect society, therefore, strict security measures are required to safeguard the grid. As a result, cybersecurity in the smart grid has become one of the most important research problems recently.

The associate editor coordinating the review of this manuscript and approving it for publication was Amedeo Andreotti.

To interrupt the normal safe operation of a power grid or gain financial advantage, cyber attackers target different elements of cyber resiliency to manipulate the data being communicated for power system operation and control. These elements include data confidentiality, data integrity, and data availability. Several prevention methods have been implemented and investigated by researchers to protect the network devices and databases from cyber intruders. For an instant, Suo *et al.* [1] investigated the latest cyber attack prevention technologies inclusively protecting sensor data, communicational devices security using encryption mechanisms and cryptographic algorithms. Mehrdad *et al.* [2] classified cyber attacks into two groups naming direct and indirect cyber attacks and further sub-categorized the direct cyber attacks into four sub-groups. Among them, data intrusion attacks are considered as the most common group of cyber attacks and its most significant attack type is Denial of services (DoS). In these attacks, to disrupt the normal trend of services, the adversary introduces artificial loads to the main service source and causes disruptions to the normal legitimate service. Most current DoS attacks are distributed (DDoS) where attackers initiate attack from several

adversaries simultaneously. Thus, detection and prevention of attack from one node will not stop the attack and make it more complicated to differentiate between legitimate and artificial service requests.

To enhance the security of the smart grid, DDoS attacks can be detected by analyzing the patterns of network data. Automatic analysis and detection methods enable in time response and preventive measures which can significantly reduce the damage. However, automatic prediction of DDoS attacks is a challenging problem. The accuracy of a DDoS attack prediction is the most critical factor for timely prevention of the attacks. To enhance the accuracy, the prediction system must learn important features from the network packets in an efficient manner. This challenge can be tackled by employing multiple learning models to enhance the prediction accuracy. However, this introduces another challenge of the automatic unification of multiple learning models. Therefore, to tackle these challenges simultaneously, we propose an automatic and efficient method for increasing the accuracy of DDoS attack predictions by employing multiple learning models. We exploit multilevel shallow and deep auto-encoders for learning rich features. For this purpose, we employ the Marginalized Stacked De-noising Auto-encoders [3] in our work due to their improved training efficiency and high accuracy. Features from a hierarchy of deep auto-encoders are unified in a weighted fashion via an efficient Multiple Kernel Learning (MKL) based on Dimensionality Reduction (MKLDR) algorithm [4]. The MKLDR algorithm is effective for combining very high dimensional features and learning a low dimensional space for classification. Our proposed method is generic and applicable in many supervised learning problems that involve automatic model fusion. We evaluate our method for DDoS attack prediction in this paper.

Following are the main contributions of this work:

- For enhanced feature encoding, we propose multiple levels of shallow and deep auto-encoders learned in an unsupervised fashion from the available training data. Multi-level auto-encoders have not been previously explored for the DDoS detection problem.
- By unifying the encoded features from all levels we learn the final more accurate detection model. For this purpose, we propose to use multiple kernel learning which automatically takes care of the relative importance of different auto-encoder based features. A small unified kernel is obtained which improves the efficiency of the detection tasks in the testing phase.
- To the best of our knowledge, DDoS detection in smart grid network using multilevel auto-encoders and MKL has not been explored previously in the literature. The proposed approach is extensively evaluated using benchmark datasets for DDoS detection in the smart grid network. The results are compared to six DDoS detection algorithms in terms of classification accuracy. Results show that the proposed method outperforms the compared algorithms.

The rest of the paper is organized as follows. Section 2 provides a literature review of the machine learning methods for DDoS detection in the smart grid network. Section 3 describes the problem statement, deep learning, and MKL. We first briefly discuss autoencoders (SDA, MDA, MSDA) and MKL for Dimensionality Reduction (MKLDR) algorithm. Next, we discuss our feature encoding strategy and ensemble model learning. Then, the overall algorithm for DDoS attack detection is presented. Section 4 presents our experimental evaluation of the proposed method. Two datasets available publicly are used to test our model. The experimental set-up and parameter choice are briefly discussed. Finally, results, comparison with previous methods and analysis are presented at the end. Section 5 presents the conclusion of our work and discussion for future research direction.

II. LITERATURE OVERVIEW

In this Section, we provide a review of pertinent literature on DDoS attack detection in smart grid networks using machine learning techniques. We summarize these techniques in Table 1. The datasets collected and used for DDoS attack detection in the smart grid network are also summarized. Most of the previous methods use shallow learning techniques or a combination of linear and non-linear methods to achieve better results. For example, Aamir and Zaidi [5] classified DDoS attacks using supervised machine learning technique including Random Forests (RF), K-Nearest neighbors (KNN) and Support Vector Machines (SVM). Wang *et al.* [6] gather attacker information by introducing honeypots in advance metering infrastructure(AMI) of the smart grid network and analyze the interaction between attacker and defender using Bayesian-Nash equilibrium to apply defense strategy accordingly. Diovu and Agee [7] prevent and mitigate DDoS attack impacts by reducing data computational burden of AMI using a firewall integrated with the cloud computing-based processing method. Srikantha and Kundur [8] proposed a collaborative reputation topology configuration based on the auto-healing method for the stability of the overall power network, while one node of the network is under attack. Varalakshmi and Selvi [9] detects and discards false malicious requests using information divergence scheme.

Specifically used machine learning algorithms for DDoS attack detection are Artificial Neural Networks (ANN), K-nearest neighbor, Support vector machine(SVM), decision tree and Naive Bayes. Generally, first, filtered network data is stored in a database. Next, the normalization of extracted features from the stored dataset for a stable training process by machine learning algorithms is achieved. In the end, this trained model is used with data packets of a real-time network for the classification of DDoS attacked and legitimate packets for further processing. Kumar and Selvakumar [10] used multiple backpropagation models for basic results. Q-statistics techniques along with Weighted Majority Voting and Weighted Product Rule are used for selecting best back

TABLE 1. Literature review of DDoS attack detection techniques using machine learning algorithms.

Author	Year	Learning method	Dataset Used
Muhammad et al. [5]	2019	kNN, SVM and RF models.	CICIDS2017
Wang et al. [6]	2017	Honeypot game theory, Bayesian-Nash equilibrium.	AMI data.
Diovu et al. [7]	2017	Cloud-based algorithms.	Attack vector, AMI traffic.
Kundur at al. [8]	2015	Collaborative reputation-based topology, Nash Equilibrium routing topology, Game theory	MATLAB simulation based data.
Selvi at al. [9]	2013	Hierarchical Broker Architecture, Kullback–Leibler Divergence.	Private Network traffic data.
Selvakumar et al. [10]	2011	RBPBoost classification algorithm, Back propagation, Weighted Majority Voting.	KDD Cup, DARPA 1999, DARPA 2000 datasets.
Wei et al. [11]	2009	Fourier to time reconstruction algorithm.	NS2 simulation data.
Poggi et al. [12]	2008	Element wise learning, Message evaluation algorithm, Naive Bayes method.	Online OmNET++ simulator based data.
Huang at al. [13]	2007	Reinforcement learning, source IP addresses monitoring, Hidden Markov Models (HMM).	DARPA 99 dataset.
Stefan at al. [14]	2007	ANNs, Extended back-propagation algorithm	Different levels of network stack data using network emulator.
Shon et al [15]	2005	Genetic Algorithm(GA), Enhanced Support Vector Machine (SVM).	1999 DARPA IDS datasct by MIT Lincoln Lab.
Symeon at al. [16]	2002	Hybrid perception based back propagation NN, Fuzzy ARTMAP.	DARPA 98, 99 datasets.

propagation model used initially, to enhance classification accuracy. However, their technique requires a manual weight setting which may not be accurate. Lu *et al.* [11] proposed a DDoS attack detection method where the service source sends pair of probes to service request node, and verify the legitimacy of request using the gap between probes using Fourier to time reconstruction algorithm. Berral *et al.* [12] used the separate networks to collect data traffic information from each node independently and then trained each node of network with Naive Bayes algorithm to demean DDoS attack impact on network by increasing efficiency of DDoS attack detection time. According to Xu *et al.* [13], majority of source IP addresses are new to target during DDoS attack. They gathered data traffic information using a source IP addresses and then applied reinforcement learning with Hidden Markov Models (HMM) for DDoS attack and suspicious nodes detection. HHM is used for probability estimation based on an observed sequence from newly added IP addresses to place detection agents near to suspicious nodes. Stefan and O'Brien [14] gave attention to utilize features of network data traffic flow and network resources to entertain the majority of legitimate user requests. Shon *et al.* [15] used Genetic Algorithm(GA) for features selection from maximum available fields of network data traffic and then Support Vector Machine (SVM) for classification of legitimate and DDoS infected packets. Manikopoulos and Papavassiliou [16] used statistical Klomogrov-Smirnov test to fetch similarities from network data measurements. After that, they applied five distinct neural network techniques for classification purposes. Backpropagation and hybrid perception based back propagation neural network techniques achieve the highest classification accuracy than others. [10], [14], [16] specifically emphasis for DDoS attack detection using network data from each network packet with neural network.

A brief look at the literature on DDoS attack detection in the smart grid network reveals that the best performing methods include the ones that use neural network-based modeling strategies. However, DDoS attack detection has not been previously explored using deep auto-encoder models. Therefore, in this study, our focus is on learning multiple levels of representations using auto-encoders and then combining these representations using the MKL algorithm. Our proposed method is simple yet more powerful as compared to the previous methods that use linear modeling techniques.

III. PROPOSED METHOD

This section describes the proposed method for DDoS attack detection in the smart grid network. The overall framework of the proposed method for learning the DDoS attack detection model in the smart grid network is illustrated in Figure 1. First, the DDoS attack detection in the smart grid network problem is formally defined. A brief introduction about the auto-encoders and Marginalized De-noising Auto-encoders is presented. After this, the theory of MKL is introduced and the popular MKL algorithm called Multiple Kernel Learning for Dimensionality Reduction (MKLDR) algorithm is summarized. Finally, the overall detection algorithm is presented.

A. PROBLEM STATEMENT

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N} \in \mathbb{R}^{d \times N}$ be the labelled training data containing d dimensional feature vectors of N different service requests. Each vector $\mathbf{x_i} \in \mathbb{R}^{d}$ contain features about customers such as source IP/port, destination IP/port, payload

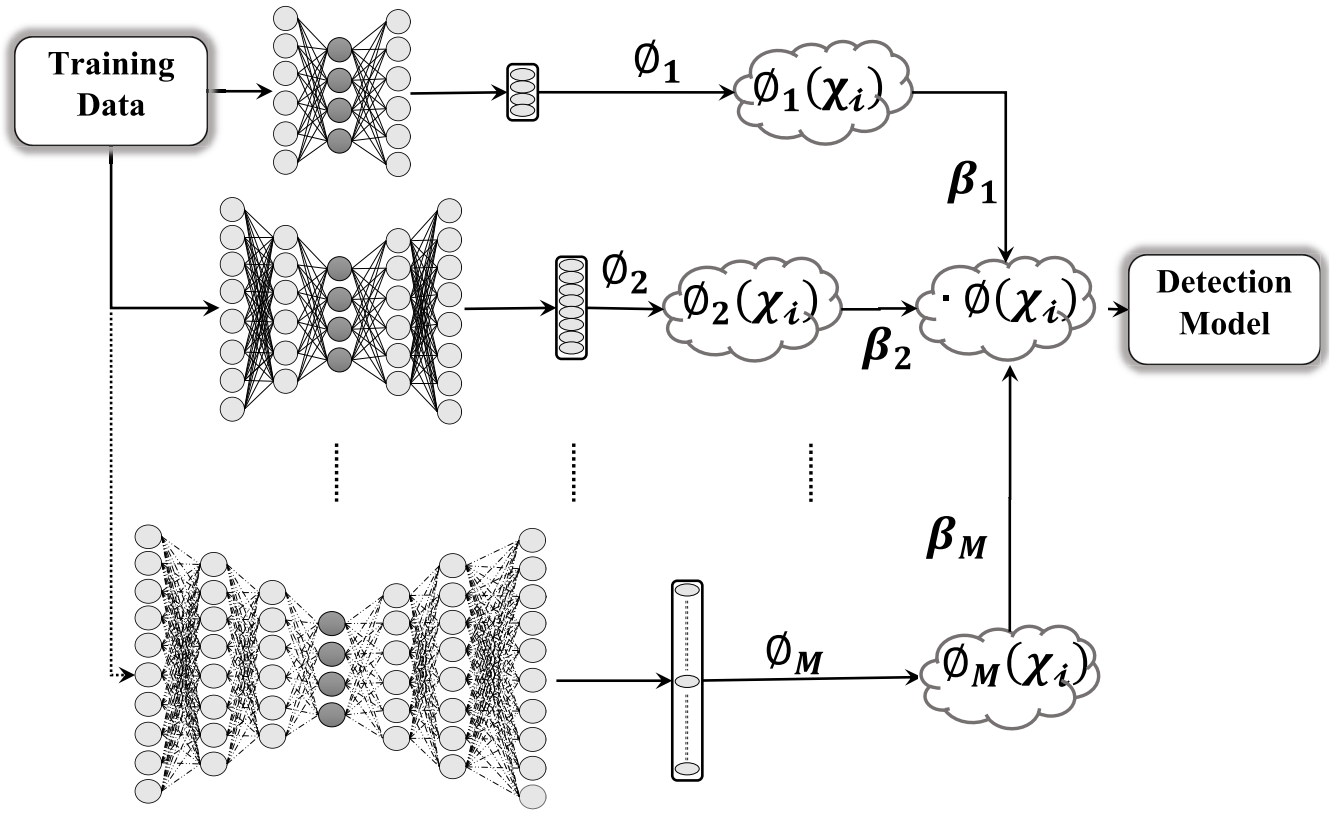


FIGURE 1. Illustration of the proposed method for learning a DDoS detection model: We first learn multilevel of shallow and deep auto-encoders in an unsupervised fashion from the available training data. Next, we generate features for every training data sample by encoding them from the hierarchy of the learned auto-encoders. In the next stage, the features are projected to a kernel space where they can be automatically combined in a weighted fashion using MKL. A unified kernel is computed which is used to compute a DDoS detection model in a supervised fashion. The final detection model is used for classification between the infected and non-infected data during test time.

bytes, packet pay size, packet reset average, destination to source packets size, and idle time etc. We also have access to the binary labels $\mathbf{y} = (y_m)_{m=1}^{N}$ also referred to as indicator variables that convey the information whether service request m is a legitimate ($y_m = 1$) or illegitimate ($y_m = 0$). The problem of DDoS attack detection involves learning a detection model $\mathbf{Z}$ from the labeled training data $\mathbf{X}$ and subsequently estimating the label y_t of a test service requests feature vector $\mathbf{x}_t$ by using the learned model.

To tackle the above problem we propose to exploit deep auto-encoders for feature learning and MKL framework for detection model learning and classification. More specifically, we first train multiple deep auto-encoders to learn rich features from training data in an unsupervised manner. Next, the features are automatically combined using automatically learned weights by MKL. The detection model obtained as a result of the MKL algorithm is then used to classify the test samples.

B. AUTO-ENCODERS

There are various categories of auto-encoders proposed previously in the deep learning literature [17]–[22]. Here we provide a brief description of how a conventional auto-encoder is used to learn features from raw data. Consider the training dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N} \in \mathbb{R}^{d\times N}$ introduced earlier. An auto-encoder is trained using a backpropagation algorithm by setting the inputs equal to the targets which are the input samples itself. In other words the purpose of the encoder is to learn the function $h_{W,b}(\mathbf{x}_i) \approx \mathbf{x}_i$ which means this encoder is learning approximations to identity function, so that the output $\hat{\mathbf{x}}_i$ is equal to $\mathbf{x}_i$.

Assume the dimensionality d of the input feature vectors $\mathbf{x}_i$ is 400 and the hidden layer units in L_2 are 200. As the numbers of hidden units are 200, it only learns the vector of these hidden units, and then reconstructs the 400-feature vector input $\mathbf{x}_i$. The reconstruction task becomes very difficult in case of a random input $\mathbf{x}_i$, which means each $\mathbf{x}_i$ is not depending on the same features. But if some features x^i of $\mathbf{x}_i$ are same or linked with each other, this algorithm will discover them. The above statements relied on a small number of hidden layer units. But interesting structures can still be discovered, if hidden units are larger in number even lager then the input, by merging other constraints to network. In the case of sigmoid activation function, when the output value is near to 1, a neuron can be used as active, and when output value is near 0, a neuron can be treated as inactive.

Normally neurons are in an inactive state. Similarly, in the case of *tanh* activation function, the state of neurons is normally inactive when the output value is near to -1.

Let $a_j^{(2)}(\mathbf{x}_i)$ represents the activation of hidden unit, the average activation function for hidden unit j is $\hat{\rho} = \frac{1}{N}\sum_{i=1}^{N}[a_j^{(2)}(\mathbf{x}_i)]$. Applying constraint ρ (sparsity parameter) $\hat{\rho} = \rho$, usually value of ρ is near to zero like $\rho = 0.09$ etc. This means, average activation of each hidden neuron j must be near to 0 to satisfy this constraint. For this purpose the objective function is $\sum_{j=1}^{s_2} \rho log\frac{\rho}{\hat{\rho}_j} + (1-\rho) log\frac{1-\rho}{1-\hat{\rho}_j}$, where s_2 = the number of units in the hidden layer. According to Kullback-Leibler ($\mathcal{KL}$) divergence concept, it can also be written as $\sum_{j=1}^{s_2} \mathcal{KL}(\rho||\hat{\rho}_j)$. Here the ($\mathcal{KL}$) is basically the divergence between a Bernoulli random variable with means ρ and $\hat{\rho}_j$.We can easily find the differences between two distributions using $\mathcal{KL}$ divergence. $\mathcal{KL}$ divergence become zero if $\hat{\rho}_j = \rho$. Therefore the cost function is:

$$j_{sparse}(\mathbf{W}, b) = j(\mathbf{W}, b) + \beta \sum_{j=1}^{s_2} \mathcal{KL}(\rho||\hat{\rho}_j) \tag{1}$$

Here, sparse penalty term weight is controlled by β. Average activation of unit j depends upon $\hat{\rho}_j$ and hidden unit activation depends on factor $\mathbf{W}$, b.

The above cost function is optimized using the backpropagation algorithm to learn the parameters W and b. In the case of a small dataset, average activation can be found by setting all forward passes to the training dataset. Then this computed activation can be used to perform backpropagation for other data samples. When a dataset is large, computer memory might become an issue, therefore, forward passes have to be set one by one, and in the end, their activations should be summed up to compute $\hat{\rho}_j$.

C. STACKED DE-NOISING AUTO-ENCODER (SDA)

As previously explained, an auto-encoder consists of two parts, an encoder $h(.)$ that is used to map an input $\mathbf{x}_i \in \mathbb{R}^d$ to some hidden representation $h(\mathbf{x}_i) \in \mathbb{R}^{d_h}$, and a decoder $g(.)$ that is used to map this hidden representation back to a reconstructed version of $\mathbf{x}_i$, such that $g(h(\mathbf{x}_i)) \approx \mathbf{x}_i$. The reconstruction error is expressed using some loss function $l(\mathbf{x}_i, g(h(\mathbf{x}_i)))$ and is minimized to learn the parameters of the auto-encoder. The loss function can take different forms including the squared error loss or Kullback-Leibler ($\mathcal{KL}$) divergence. De-noising auto-encoders work on the principal that the input samples are corrupted slightly before mapping them into the hidden layer representation. Their training involves reconstruction of the (or denoising) of the actual input $\mathbf{x}_i$ from its corrupted version $\tilde{\mathbf{x}}_i$ by minimizing $l(\mathbf{x}_i, g(h(\tilde{\mathbf{x}}_i)))$. Various forms of corruptions exist including additive isotropic Gaussian noise or binary masking noise. Binary masking noise is also popular which sets a portion of the features for each input sample equal to zero. Several de-nosing auto-encoders can be stacked (Stacked De-noising Auto-encoders (SDA) [23]) for learning deep feature representation consisting of multiple layers.

Although SDAs [23] can be used to learn rich features for classification, they have several disadvantages. For example, their training time is slow due to the stochastic gradient descent based back-propagation algorithm. SDAs also contain multiple hyper-parameters such as learning rate, number of epochs, noise ratio, mini-batch size and network structure, which require tuning using a validation dataset. This can add further time to the already slow training process. Moreover, the optimization is non-convex and initialization plays a key role in final the results.

D. MARGINALIZED DE-NOISING AUTO-ENCODER (MDA)

Recently, Chen *et al.* [3] proposed an improved version of the SDA called Marginalized SDA (MDA) to improve the training time significantly. The MDA algorithm reduces to a closed form solution and thus does not require a backpropagation algorithm to learn the network parameters. This makes MDA computations more efficient as compared to conventional SDA. Furthermore, several MDAs can be stacked together to generate deep feature representation. Moreover, the classification accuracy of the MDA has been found to be similar to that of the SDA [3]. Therefore, inspired by these advantages, we employ the MDA algorithm to learn robust features for DDoS detection in this study.

The basic building block of the MDA is a single layer de-noising auto-encoder. Each input sample $\mathbf{x}_i$ from the training dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N} \in \mathbb{R}^{d\times N}$ is corrupted by randomly removing features (setting them to zero). Specifically, a feature is given a value of 0 with probability $p \geq 0$.

Let the corrupted version of $\mathbf{x}_i$ is denoted by $\tilde{\mathbf{x}}_i$. In contrast to the two step encoder and decoder mechanism in SDA, MDA reconstructs from the corrupted input samples using a single parameter matrix $\mathbf{W} : \mathbb{R}^d \rightarrow \mathbb{R}^d$, that minimizes the following reconstruction loss:

$$\frac{1}{2N}\sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_i||^2. \tag{2}$$

Note, to incorporate the bias term in the formulation a constant feature is added in the input feature vector, $\mathbf{x}_i = [\mathbf{x}_i; 1]$. This includes the bias into the parameters $\mathbf{W} = [\mathbf{W}, \mathbf{b}]$ and the bias is not corrupted during MDA learning.

The solution to (2) depends on the random corruptions of the individual features of the input. To reduce the variance, MDA performs multiple iterations over the training data and use different corruption in each iteration. Then, the parameter matrix $\mathbf{W}$ is learned in a way that the overall squared loss is minimized:

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2RN}\sum_{j=1}^{R}\sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_{i,j}||^2. \tag{3}$$

where $\tilde{\mathbf{x}}_{i,j}$ denotes the jth corrupted copy of the original input $\mathbf{x}_i$.

Denote the R-times repeated version of the training data matrix $\mathbf{X}$ as $\bar{\mathbf{X}} = [\mathbf{X}, \ldots, \mathbf{X}]$. Similarly, the corrupted copy

of $\bar{\mathbf{X}}$ is denoted as $\tilde{\mathbf{X}}$. The overall loss in (2) can now be written as:

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2RN}\mathbf{tr}\left[\left(\overline{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\right)^T\left(\overline{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}}\right)\right] \tag{4}$$

The closed form solution to (4) can be obtained by using least squares algorithm [24]:

$$\mathbf{W} = \mathbf{P}\mathbf{Q}^{-1} \quad where\ \mathbf{Q} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\ and\ \mathbf{P} = \overline{\mathbf{X}}\tilde{\mathbf{X}}^T \tag{5}$$

The larger R is, the more corruption are averaged. Ideally, $R \to \infty$ means that to learn the parameter matrix $\mathbf{W}$ we should use infinitely many versions of the corrupted data. The matrices $\mathbf{P}$ and $\mathbf{Q}$ converge to their expected values R and tends to infinity by using the weak law of large numbers. Therefore, the expectations of $\mathbf{Q}$ and $\mathbf{P}$ is derived and expressed as the parameter matrix $\mathbf{W}$ as:

$$\mathbf{W} = E[\mathbf{P}]E[\mathbf{Q}]^{-1}. \tag{6}$$

To compute the expectations of these two matrices let

$$E[\mathbf{Q}] = \sum_{i=1}^{n} E[\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T]. \tag{7}$$

A non-diagonal element in the matrix $\tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T$ remains uncorrupted if the two features α and β survive the corruption with probability $(1-p)^2$. This is done for the diagonal elements with a probability $1-p$. Next, the vector $\mathbf{q} = [1-p, \ldots, 1-p, 1]^T \in \mathcal{R}^{d+1}$, is defined in which $\mathbf{q}_\alpha$ denotes the probability of a feature α which stays uncorrupted. Moreover, $\mathbf{S} = \mathbf{X}\mathbf{X}^T$ is defined as the scatter matrix of the uncorrupted input features and the expectation of the matrix $\mathbf{Q}$ is expressed as:

$$E[\mathbf{Q}]_{\alpha,\beta} = \begin{cases} \mathbf{S}_{\alpha\beta}\mathbf{q}_\alpha\mathbf{q}_\beta & if\ \alpha \neq \beta \\ \mathbf{S}_{\alpha\beta}\mathbf{q}_\alpha\mathbf{q}_\beta & if\ \alpha \neq \beta \end{cases} \tag{8}$$

In a similar way, the expectations of $\mathbf{P}$ can also be obtained in closed-form as $E[\mathbf{P}]_{\alpha\beta} = \mathbf{S}_{\alpha\beta}\mathbf{q}_\beta$. With the help of these matrices of expectations, the re-constructive parameter matrix $\mathbf{W}$ is computed directly without even constructing a single corrupted input $\tilde{\mathbf{x}}_i$. Thus, this algorithm is termed as Marginalized De-noising Auto-encoder (MDA).

The MDA algorithm has multiple advantages over the traditional DAs: For example, MDA needs only a single iteration over the data to compute the matrices $E[\mathbf{Q}]$,$E[\mathbf{P}]$. Moreover, the optimization problem is convex and has a globally optimal closed-form solution.

Similar to SDAs, non-linearity can also be incorporated in MDA using a non-linear transfer function $h(.)$. More specifically, the output of each MDA is passed through a non-linear transfer function for non-linear mapping. In this manner, the parameter matrix $\mathbf{W}$ will be able to learn non-linear features. Several types of non-linear transfer functions can be used such as sigmoid function, hyperbolic tangent and the recently proposed rectifier linear unit [25]. In this study, we use the tanh() transfer function.

E. FEATURE GENERATION USING MARGINALIZED STACKED DE-NOISING AUTO-ENCODER (MSDA)

In this paper, we stack multiple layers of MDAs by connecting the output of the $(t-1)^{th}$ MDA (after the transfer function) to the input of the t^{th} MDA. The output of the t^{th} MDA is $\mathbf{h}_t$ and the original input is denoted to be $\mathbf{h}_0 = \mathbf{x}$. To stacke multiple MDAs, the training is performed in a greedily layerwise fashion i.e. each parameter matrix $\mathbf{W}_t$ is learned to reconstruct the previous layer output $\mathbf{h}_{t-1}$ using all the possible corruptions and the output of the t^{th} layer becomes $\mathbf{h}_t = \tanh(\mathbf{W}_t\mathbf{h}_{t-1})$. The stacked de-noising algorithm is referred to as Marginalized Stacked De-noising Auto-encoder (MSDA) [3]. The key aspects which are responsible for the success of the SDAs include its non-linearity and ability to learn deep features. MSDA framework also included these capabilities in addition to its fast training time. Once the auto-encoder is trained, we use it for non-linear feature generation. To generate feature representation for a sample $\mathbf{x}_i$ we first pass it through the learned MSDA with L layers. We then concatenate all the activations $\mathbf{h}_0 - \mathbf{h}_L$ into a single feature vector denoted as $\mathbf{f}_i \in \mathbb{R}^{(d*L)}$.

To achieve a multilevel representation, we train a mixture of shallow (having few layers) and deep (having many layers) MSDAs. Specifically, we train M number of different MSDAs having a different numbers of layers to generate M multiple feature representations of a sample $\mathbf{x}_i$ (See Fig. 1). Thus the training data $\mathbf{X}$ is transformed to $\mathbf{F}^m = \{\mathbf{f}_i^m\}_{i=1}^N$ under the mth representation. Such multilevel representation is more discriminative than the single level representation.

IV. MULTIPLE KERNAL LEARNING (MKL)

Several previous works like [26]–[30] shows that the accuracy of a classifier can be greatly enhanced by fusing multiple features. Therefore, in this work, we propose to use MKL to learn a unified detection model from the multilevel auto-encoder features. For this purpose, we choose an efficient MKL algorithm called Multiple Kernel Learning for Dimensionality Reduction (MKLDR) [4] algorithm.

A. THE MULTIPLE KERNEL LEARNING FOR DIMENSIONALITY REDUCTION (MKLDR) ALGORITHM

The MKLDR algorithm [4] optimally combines several feature representation into a unified feature representation. MKLDR gives a general dimension reduction framework for features representation of data using multiple kernels. MKLDR effectively works in learning a detection model for supervised learning problems. Here we gave a description of the MKLDR algorithm. First, the process of making base kernels from multiple features representations are explained. Next, the process of learning the ensemble kernel in a discriminative dimensionality reduction framework is explained.

Let M be the number of MSDAs so that each sample in our training data has M different feature representations i.e the training data $\mathbf{X}$ is transformed to $\mathbf{F}^m = \{\mathbf{f}_i^m\}_{i=1}^N \in \mathbb{R}^{d\times N}$

under the mth deep representation. First, each feature representation is described as a kernel matrix. For this purpose, a distance matrix $\mathbf{D}_m \in \mathbb{R}^{N\times N}$ is constructed $\mathbf{D}_m(i,j) = d_m(\mathbf{f}_i^m, \mathbf{f}_j^m)$; where d_m is the distance under the m feature representation. The groupwise distances of data samples are converted to kernal matrices [31], [32] using the Gaussian kernel as $\mathbf{K}_m(i,j) = \exp\left(\frac{-\mathbf{D}_m^2(\mathbf{f}_i^m, \mathbf{f}_j^m)}{\sigma_m^2}\right)$, where σ_m is the Gaussian scale factor. Next, for M kernel matrices $\{\mathbf{K}_m\}_{m=1}^M$, the MKLDR algorithm learns an ensemble kernel in a weighted fashion as:

$$\mathbf{k}(\mathbf{f}_i^m, \mathbf{f}_j^m) = \sum_{m=1}^{M} \beta_m k_m(\mathbf{f}_i^m, \mathbf{f}_j^m), \quad \beta_m \leq 0, \tag{9}$$

where the weights β_m for each kernel are automatically learned in a discriminative learning framework. $\mathbf{K}_m$ may not be always positive, but by adding the smallest eigenvalue of K_m into the diagonal, this issue can be resolved [32]. In case if it's still negative, then the absolute value is added to the diagonal of $\mathbf{K}_m$. The optimal weights $\beta_1, \beta_2, \ldots . \beta_M$ can now be learned for fusing these M different types of deep feature representations. This makes the MKLDR algorithm applicable to many diverse input kernels and distance measures efficiently. Kernelization in MKLDR is nearly similar to kernel PCA in [33], but the only difference is that MKLDR uses more than one kernels $\{k_m\}_{m=1}^M$ [34].

Let $\phi : \mathbf{F}^m \rightarrow : \mathcal{F}$ is feature mapping induced by $\mathbf{K}$ such that $\mathbf{f}_i^m \rightarrow \phi(\mathbf{f}_i^m), \quad for\ i = 1, 2, \ldots . N$. The MKLDR uses the kernel trick to learn a projection $\mathbf{v}$ and the weights β_m simultaneously

$$\mathbf{v}^T\phi(\mathbf{f}_i^m) = \sum_{n=1}^{N}\sum_{m=1}^{M} \alpha_n\beta_m k_m(\mathbf{f}_n^m, \mathbf{f}_i^m) = \boldsymbol{\alpha}^T\mathbb{K}^{(i)}\boldsymbol{\beta} \tag{10}$$

where $\boldsymbol{\beta} = [\beta_1 \ldots . \beta_M]^T \in \mathbb{R}^M$, $\boldsymbol{\alpha} = [\alpha_1 \ldots . \alpha_N]^T \in \mathbb{R}^N$ and

$$\mathbb{K}^{(i)} = \begin{bmatrix} \mathbf{K}_1(1,i) & \ldots & \mathbf{K}_M(1,i) \\ \vdots & \ddots & \vdots \\ \mathbf{K}_1(N,i) & \ldots & \mathbf{K}_M(N,i) \end{bmatrix}$$

With (10) the constrained optimization problem for one-dimensional MKLDR is as follow:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{i,j=1}^{N} ||\boldsymbol{\alpha}^T\mathbb{K}^{(i)}\boldsymbol{\beta} - \boldsymbol{\alpha}^T\mathbb{K}^{(j)}\boldsymbol{\beta}||^2 w_{ij} \tag{11}$$

$$s.t. \sum_{i,j=1}^{N} ||\boldsymbol{\alpha}^T\mathbb{K}^{(i)}\boldsymbol{\beta} - \boldsymbol{\alpha}^T\mathbb{K}^{(j)}\boldsymbol{\beta}||^2 w'_{ij} = 1,$$
$$\beta_m \geq 0 \quad for\ \ m = 1, 2, \ldots, M. \tag{12}$$

The positivity constraints in (12) are introduced so that the learned ensemble kernel $\mathbf{K}$ in MKLDR is a non-negative combination of the individual base kernels. Note that $\mathbf{v}$ denotes the one-dimensional projection consisting of the coefficients in the vector $\boldsymbol{\alpha}$ and the kernel weights $\boldsymbol{\beta}$. These two vectors contribute to the construction of the projection according to the respective strength of the input samples and the base kernels. To derive a multidimensional version of the projection $\mathbf{v}$, P coefficient vectors learned can be represented as $\mathbf{A} = [\boldsymbol{\alpha}_1\ \boldsymbol{\alpha}_2\ \ldots . \boldsymbol{\alpha}_P]$.

Given the projection matrix $\mathbf{A}$ and weight vector $\boldsymbol{\beta}$, each one dimensional projection $\mathbf{v}_i$ can be found by the respective coefficient vector $\boldsymbol{\alpha}_i$ and the kernel weight vector $\boldsymbol{\beta}$. The learned projection matrix $\mathbf{V} = [v_1\ v_2 \ldots . v_P]$ can now project samples to the P-dimensional Euclidean space where classification can be performed. Analogously to the one dimensional case, the sample projection $\mathbf{x}_i$ can be expressed as $\mathbf{V}^T\phi(x_i) = \mathbf{A}^T\mathbb{K}^{(i)}\boldsymbol{\beta} \in \mathbb{R}^P$.. The optimization problem (11) is extended for the multidimensional case as:

$$\min_{\mathbf{A},\boldsymbol{\beta}} \sum_{i,j=1}^{N} ||\mathbf{A}^T\mathbb{K}^{(i)}\boldsymbol{\beta} - \mathbf{A}^T\mathbb{K}^{(j)}\boldsymbol{\beta}||^2 w_{ij} \tag{13}$$

$$s.t. \sum_{i,j=1}^{N} ||\mathbf{A}^T\mathbb{K}^{(i)}\boldsymbol{\beta} - \mathbf{A}^T\mathbb{K}^{(j)}\boldsymbol{\beta}||^2 w'_{ij} = 1,$$
$$\beta_m \geq 0, \quad for \quad m = 1, 2, \ldots, M. \tag{14}$$

The four types of spaces involved in the MKLDR algorithm are illustrated in Fig. 1. These include the input feature space consisting of the deep features learned via MSDAs, the kernel space which consists of the input features mapped to a Reproducing Kernel Hilbert Space (RKHS) via a valid kernel and the Euclidean space where the projection model is learned.

As the direct optimization of a problem (13) is often difficult, MKLDR algorithm [4] uses an iterative two-step algorithm to optimize $\mathbf{A}$ and $\boldsymbol{\beta}$ in an alternating fashion. In every iteration, the optimization of one of $\mathbf{A}$ or $\boldsymbol{\beta}$ is performed while the other is fixed. The roles of $\mathbf{A}$ and $\boldsymbol{\beta}$ are switched. These iterations are repeated until convergence is achieved or up to some maximum number of iterations.

Optimizing $\mathbf{A}$. By fixing $\boldsymbol{\beta}$ and using the property $||u||^2 = trace(uu^T)$ for a column vector u, the optimization problem (13) is reduced to

$$\min_{\mathbf{A}} trace(\mathbf{A}^T\mathbf{S}_w^{\boldsymbol{\beta}}, A)$$
$$s.t.\ trace(\mathbf{A}^T\mathbf{S}_w^{\boldsymbol{\beta}}, \mathbf{A}) = 1 \tag{15}$$

where

$$\mathbf{S}_W^{\boldsymbol{\beta}} = \sum_{i,j=1}^{N} w_{ij}(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})\boldsymbol{\beta}\boldsymbol{\beta}^T(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})^T$$
$$\mathbf{S}_{W'}^{\boldsymbol{\beta}} = \sum_{i,j=1}^{N} w'_{ij}(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})\boldsymbol{\beta}\boldsymbol{\beta}^T(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})^T \tag{16}$$

where $\mathbf{W}$ and $\mathbf{W}'$ are the affinity matrices defined for a specific dimensionality reduction framework e.g. Linear Discriminant Analysis (LDA) and w_{ij} and w'_{ij} represents the values at the index ij. It can be observed that the problem (15) is a trace ratio optimization problem, $min_{\mathbf{A}} \frac{trace(\mathbf{A}^T\mathbf{S}_w^{\boldsymbol{\beta}}\mathbf{A})}{trace(\mathbf{A}^T\mathbf{S}_{w'}^{\boldsymbol{\beta}}\mathbf{A})}$. Following [35] and [36], a closed-form solution can be

obtained by converting (15) into a corresponding ratio trace problem, i.e., $min_{\mathbf{A}}\ trace[(\mathbf{A}^T\mathbf{S}_{w'}^{\boldsymbol{\beta}}\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{S}_{w}^{\boldsymbol{\beta}}\mathbf{A})]$.

Consequently, the columns of the optimal $\mathbf{A}^* = [\boldsymbol{\alpha}_1\ \boldsymbol{\alpha}_2\ \ldots..\boldsymbol{\alpha}_P]$, consists of the eigenvectors that correspond to the first P smallest eigenvalues of $\mathbf{S}_{w}^{\boldsymbol{\beta}}\boldsymbol{\alpha} = \lambda\mathbf{S}_{w'}^{\boldsymbol{\beta}}\boldsymbol{\alpha}$.

Optimizing $\boldsymbol{\beta}$. Next, $\mathbf{A}$ is fixed and letting $||u||^2 = u^T u$, the problem in (13) now reduces to:

$$\begin{aligned} &min_{\boldsymbol{\beta}}\ \boldsymbol{\beta}^T\mathbf{S}_w^{\mathbf{A}}\boldsymbol{\beta} \\ &s.\,t.\ \boldsymbol{\beta}^T\mathbf{S}_w^{\mathbf{A}}\boldsymbol{\beta} = 1 \\ &\quad and \quad \boldsymbol{\beta} \geq 0, \end{aligned} \tag{17}$$

where

$$\begin{aligned} \mathbf{S}_W^{\mathbf{A}} &= \sum_{i,j=1}^{N} w_{ij}(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})^T\mathbf{A}\mathbf{A}^T(\mathbb{K}^{(i)} - \mathbb{K}^{(j)}), \\ \mathbf{S}_{W'}^{\mathbf{A}} &= \sum_{i,j=1}^{N} w'_{ij}(\mathbb{K}^{(i)} - \mathbb{K}^{(j)})^T\mathbf{A}\mathbf{A}^T(\mathbb{K}^{(i)} - \mathbb{K}^{(j)}) \end{aligned} \tag{18}$$

The additional constraints $\boldsymbol{\beta} \geq 0$ make the optimization of problem (17) to be no longer a generalized eigenvalue problem. Rather, the problem now becomes a non-convex quadratically constrained quadratic programming (QCQP) problem, which is generally hard to solve. One way to solve it is to use its convex relaxation by adding the auxiliary variable $\mathbf{B}$ of size $M \times M$:

$$\min_{\boldsymbol{\beta},\mathbf{B}}\ trace(\mathbf{S}_w^{\mathbf{A}}, \mathbf{B})\ \ s.\ t.\ \ trace(\mathbf{S}_w^{\mathbf{A}}, \mathbf{B}) = 1 \tag{19}$$

$$e_m^T\boldsymbol{\beta} \geq 0, \quad m = 1, 2, \ldots.., M \tag{20}$$

$$\begin{bmatrix} 1 & \boldsymbol{\beta}^T \\ \boldsymbol{\beta} & \mathbf{B} \end{bmatrix} \succeq 0 \tag{21}$$

where $\mathbf{e}_m$ in (20) is a column vector of zeros except having a 1 at the mth index. Moreover, the constraint in (21) enforces that the square matrix should be positive semi-definite. The above problem is solved using the semi-definite programming (SDP) as explained in [4].

The algorithm of MKLDR needs an initial value for either $\mathbf{A}$ or $\boldsymbol{\beta}$ in the alternating optimization. Two different variations can be used. For example, $\boldsymbol{\beta}$ can be initialized by setting all the weight elements to 1. Similarly, $\mathbf{A}$ can also be initialized by setting it to identity matrix $\mathbf{A}\mathbf{A}^T = I$. Usually, the second initialization strategy achieves better results than the first strategy. In our experiments, we used the second initialization strategy to learn our model.

B. TEST SAMPLE CLASSIFICATION

After accomplishing the training procedure of MKLDR, we are ready to project a testing sample which is encoded from the multilevel auto-encoders, say $\mathbf{f}_t$, into the learned space of lower dimension by

$$\mathbf{x}_t \mapsto \mathbf{A}^T\mathbb{K}^{(z)}\boldsymbol{\beta}, \tag{22}$$

where $\mathbb{K}^{(z)} \in \mathbb{R}^{N\times M}$ *and* $\mathbb{K}^{(z)}(n, m) = k_m(\mathbf{f}_n, \mathbf{f}_t)$.

After the low dimensional feature representation is obtain for $\mathbf{f}_t$ we use the Nearest Neighbor (NN) classifier to estimate the label.

V. EXPERIMENTAL RESULTS

We perform experiments on two benchmark intrusion detection datasets and compare the results of the proposed algorithms in terms of DDoS detection accuracy with six recent machine learning based DDoS detection algorithms. These methods include Naive-Bayes [37], Decision Tree [37], KN [38], LSVM [38], Random Forest [39] and LSTM [39]. For the baseline, we learn a detection model using the linear SVM on the raw features of the labeled training data. The details of the datasets used in our experiments are provided below.

A. UNB ISCX INTRUSION DETECTION EVALUATION 2012 DATASET

The first publicly available dataset used in our experiments is UNB ISCX Intrusion Detection Evaluation 2012 dataset named IDE2012. We use the 11th June testbed named as IDE2012/11 and 16th June testbeds [40] named as IDE2012/16. In the IDE2012/11, there are 325,757 samples (packets) each having 204 features. Similarly, in IDE2012/16, there are 464,989 samples each having 204 features. The labels of each sample are provided as safe, unsafe, acceptable and unrated. We discretize the labels into safe (0), unrated (1) and acceptable (2). Some example features of each data packet include values such as source port, destination port, event generator, event signature, event priority, ndpi risk, ndpi detected protocol, payload bytes first, etc. Features that are not numbers were discretized. According to this dataset, 10000 packets are infected by DDoS attacks, which is 15% of total packets, showing 15% is the infection rate. Part of this dataset is shown in Table 2.

TABLE 2. Example features and their corresponding labels of four packets in the IDE2012/11 DataSet.

src port	dst port	payload bytes first	dst2src packet rate	src2dst packet rate	packet header size	ndpi detection	ndpi risk
64397	80	0	0	0	52	0	Safe
4585	110	0	933	466	48	1	Unsafe
4948	22	0	12	4	40	2	Acceptable
1555	80	0	0	0	40	3	Unrated

To test our model's ability to learn from various amounts of data, we also randomly sample 10,000 samples from each dataset. In addition to the two-class classification settings safe (0), unsafe (1), we also evaluate our model on the four class classification settings where the labels correspond to safe (0), unsafe (1), acceptable (2) and unrated (3). The details and names for these splits of the datasets are shown in Table 3.

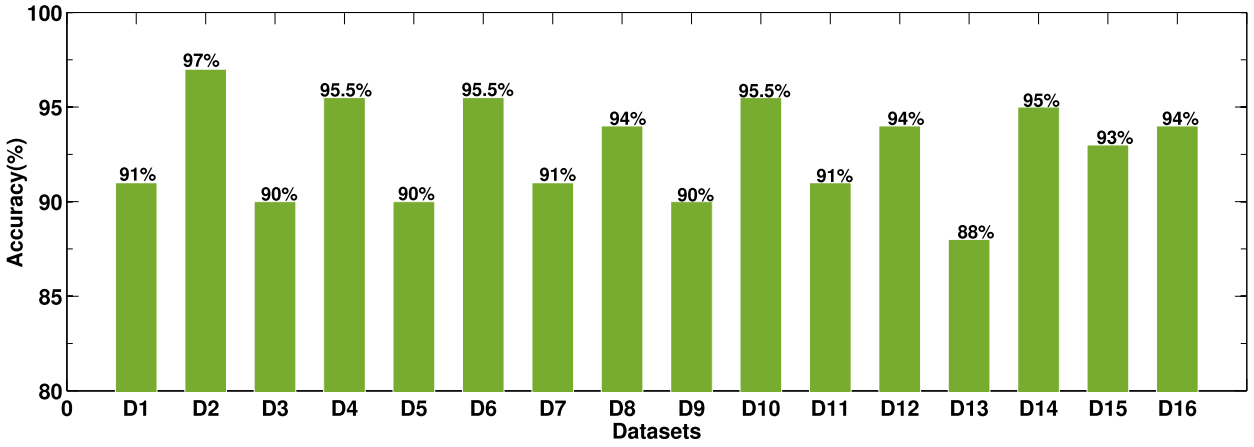


FIGURE 2. The Average accuracy of the proposed method on the 16 datasets.

TABLE 3. Details of the packets, features, and subsets generated from the IDE2012 Dataset.

Name	Dataset	Packets	Features	Labels
D1	IDE2012/11	325,757	204	0,1
D2	IDE2012/11	10,000	204	0,1
D3	IDE2012/11	325,757	204	0,1,2,3
D4	IDE2012/11	10,000	204	0,1,2,3
D5	IDE2012/16	464,989	204	0,1
D6	IDE2012/16	10,000	204	0,1
D7	IDE2012/16	464,989	204	0,1,2,3
D8	IDE2012/16	10,000	204	0,1,2,3

TABLE 4. Details of the packets, features and subsets generated from the UNSW-NB15 dataset.

Named	Dataset	Packets	Features	Labels
D9	UNSW NB15-1	700,001	49	0,1
D10	UNSW NB15-1	10,000	49	0,1
D11	UNSW NB15-2	700,001	49	0,1
D12	UNSW NB15-2	10,000	49	0,1
D13	UNSW NB15-3	700,001	46	0,1
D14	UNSW NB15-3	10,000	46	0,1
D15	UNSW NB15-4	440,044	46	0,1
D16	UNSW NB15-4	10,000	46	0,1

B. UNSW-NB15 DATASET

The next publicly available dataset that we used in our experiments is the UNSW-NB15 [41] dataset. The Australian Center for Cyber Security (ACCS) used the IXIA Perfectstorm tool to create the UNSW-NB15 dataset. There is a total of 7,00,001 packets (samples) each having 49 features. Some example features include source IP, source port, destination IP, destination port, transaction protocol, state, duration, source jitters (mSec), destination jitters (mSec), record start time, record last time and attack category, etc. For our work, we convert the non-number feature into discrete features. The labels are provided as non-attack and attack. We discretize the labels as an attack (0) and non-attack (1). Again, to push our model limits divided 4 sets of this dataset as shown in Table 4.

VI. EXPERIMENTAL SETUP

In total, we have 16 datasets named D1 to D16. To evaluate our model, we randomly divide these datasets into two portions, one having 80% data, and the other one with 20% data of this dataset. 80% of data is then used to train our model, and 20% is for testing purposes. This process is repeated 10 times and average accuracy is reported for 10 experiments.

To analyze the performance of the proposed algorithm, we used the following performance indicators. *TN* (True Negative) is used for the amount of normal data detected as normal. *FN* (False Negative) is used for the amount of normal data detected as infected. *TP* (True Positive) is used for the amount of infected data detected as infected. *FP* (False Positive) is used for the amount of infected data detected as normal. These quantities are used to measure the $Accuracy = \frac{TN+TP}{TN+FN+TP+FP}$ of the proposed algorithm.

The proposed algorithm includes the parameters of the MSDA and the parameters of the MKLDR algorithm. For feature learning the parameters include a number of deep MSDAs M, corruption probability p and the number of layers in each MSDA L. We use $M = 9$ MSDAs in our experiments. The number of layers in each MSDA is selected as $L_m = [1, 3, 5, 7, 9, 11]$. The corruption probability p_m for each MSDA is set from the set 0.1,0.2,...,0.5. Note that these parameters can also be further tuned in a cross-validation framework for more improved results. For MKL the MKLDR algorithm includes the Gaussian scale factor for each kernel σ_m, the choice of dimensionality reduction algorithm to compute the affinity matrices $\mathbf{W}$, $\mathbf{W}'$ and the dimensionality of the low dimensional space. In MKLDR the parameter σ_m

TABLE 5. Average accuracy and standard deviations of 10-fold cross validation experiments.

Datasets	LSTM [39]	Random Forest [39]	Naive-Bayes [37]	Decision-tree [37]	KNN [38]	LSVM [38]	**Proposed**
D1	90.5 ± 1.3	89.7 ± 1.7	88.1 ± 2.3	89.9 ± 1.6	90.0 ± 1.9	90.1 ± 2.5	$\mathbf{91.0 \pm 2.0}$
D2	96.3 ± 2.1	95.2 ± 2.1	93.6 ± 1.6	95.8 ± 1.3	95.9 ± 2.3	92.8 ± 1.9	$\mathbf{97.0 \pm 2.8}$
D3	88.7 ± 1.5	86.1 ± 1.8	88.5 ± 1.2	87.3 ± 1.1	88.1 ± 1.5	86.4 ± 2.0	$\mathbf{90.0 \pm 1.9}$
D4	94.7 ± 1.9	89.8 ± 1.3	93.3 ± 1.0	94.1 ± 1.4	92.3 ± 2.8	90.5 ± 1.9	$\mathbf{95.5 \pm 2.7}$
D5	89.1 ± 2.0	88.2 ± 1.8	87.5 ± 1.1	86.8 ± 2.2	88.3 ± 1.9	85.5 ± 1.5	$\mathbf{90.0 \pm 1.1}$
D6	94.2 ± 1.1	93.3 ± 2.1	92.6 ± 0.7	91.8 ± 1.7	93.4 ± 2.5	89.7 ± 1.9	$\mathbf{95.5 \pm 2.7}$
D7	90.2 ± 2.0	88.2 ± 1.7	85.5 ± 1.2	87.8 ± 1.3	84.7 ± 1.9	88.3 ± 2.3	$\mathbf{91.0 \pm 2.0}$
D8	93.6 ± 1.3	87.7 ± 1.9	90.2 ± 1.7	89.5 ± 2.1	93.3 ± 1.5	90.2 ± 2.8	$\mathbf{94.0 \pm 2.3}$
D9	88.7 ± 1.6	86.7 ± 1.1	88.1 ± 1.2	84.5 ± 1.6	79.3 ± 2.3	82.3 ± 2.5	$\mathbf{90.0 \pm 1.9}$
D10	94.1 ± 2.0	89.6 ± 1.3	93.6 ± 2.1	87.3 ± 1.9	92.2 ± 2.3	84.1 ± 1.5	$\mathbf{95.5 \pm 2.6}$
D11	90.1 ± 1.2	87.2 ± 1.6	84.3 ± 1.8	84.9 ± 1.0	80.9 ± 1.5	89.2 ± 1.9	$\mathbf{91.0 \pm 1.9}$
D12	93.1 ± 1.6	88.7 ± 1.8	90.2 ± 1.5	92.1 ± 1.7	88.8 ± 2.5	87.3 ± 2.3	$\mathbf{94.0 \pm 2.3}$
D13	86.2 ± 2.7	80.4 ± 1.1	81.6 ± 1.5	83.8 ± 1.8	85.7 ± 2.3	84.2 ± 2.5	$\mathbf{88.0 \pm 1.2}$
D14	93.3 ± 1.4	90.3 ± 1.3	89.7 ± 1.2	91.6 ± 1.7	87.8 ± 2.5	90.9 ± 2.7	$\mathbf{95.0 \pm 2.5}$
D15	92.5 ± 1.9	87.7 ± 1.4	91.6 ± 1.7	86.8 ± 1.6	92.1 ± 2.3	86.5 ± 1.5	$\mathbf{93.0 \pm 2.5}$
D16	$\mathbf{95.5 \pm 2.0}$	86.2 ± 1.9	92.3 ± 2.0	92.9 ± 1.0	85.7 ± 2.5	88.9 ± 1.9	94.0 ± 2.3
Average	92.8 ± 1.6	88.5 ± 1.6	89.4 ± 1.5	89.2 ± 1.6	88.7 ± 2.2	88.0 ± 2.1	$\mathbf{93.0 \pm 2.2}$

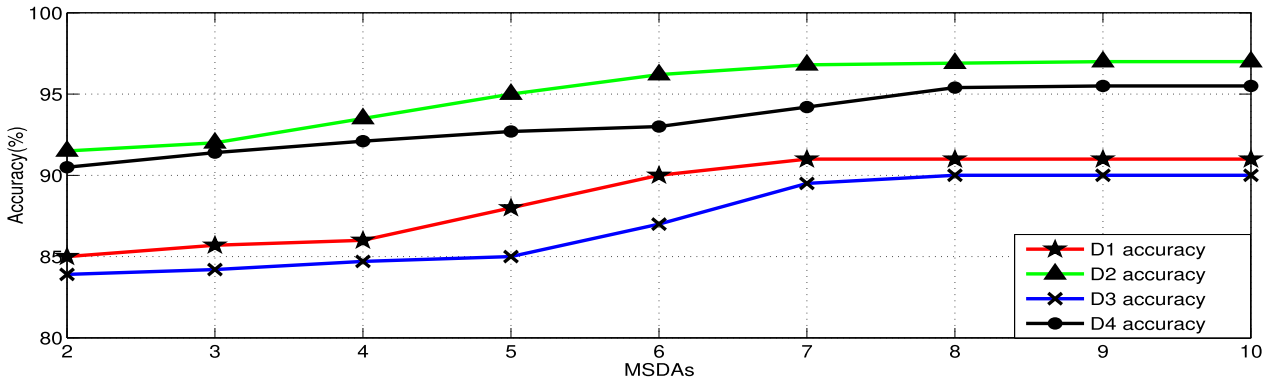


FIGURE 3. Effect on the accuracy by using a different number of MSDAs on subsets of ISCX IDE 11-June dataset. The final accuracy saturates beyond using 7 MSDAs.

is automatically tuned by the method given in [4]. For the computation of affinity matrices, we select the LDA algorithm. The parameters of the compared algorithms are set according to the recommendations of the original authors to achieve the best results. For both datasets and their sub datasets, we perform 10-fold cross-validation and the average of the 10-folds is presented as the final detection accuracy. All the experimentation were performed on a computer with 32.5 GB memory and NVIDIA Tesla V100 GPUs using Matlab implementations.

VII. RESULTS AND ANALYSIS

Figure 2 shows that the average accuracy achieved by the proposed method on the 16 datasets. Achieved accuracy with large datasets (D1,D3,...,D15) is slightly lower as compared to accuracy achieved on the smaller datasets. This is because of large number of test sets in large dataset. However, the proposed method can learn equally good features from both small and large datasets. The highest accuracy of 97% is achieved on the D2 dataset.

Table 5 shows a comparison of the average accuracy of the proposed method with six other machine learning based methods for DDoS attack detection. It can be observed that the proposed algorithm has significantly outperformed the compared algorithms. This is because our algorithm is able to learn rich features in multiple auto-encoders and then combine these representations in the kernel domain. Most of the previous methods use a single representation that is unable to achieve satisfactory accuracy. Our algorithm, on the other hand, uses multiple deep models that are able to learn useful

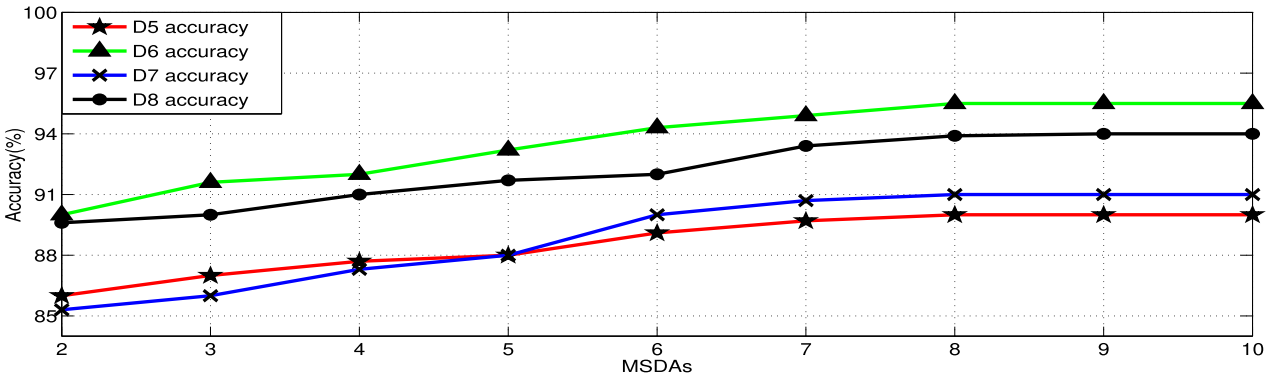


FIGURE 4. Relationship of the number of MSDAs vs the final accuracy on different subsets of ISCX IDE 16-June dataset. The final accuracy saturates beyond using 8 MSDAs.

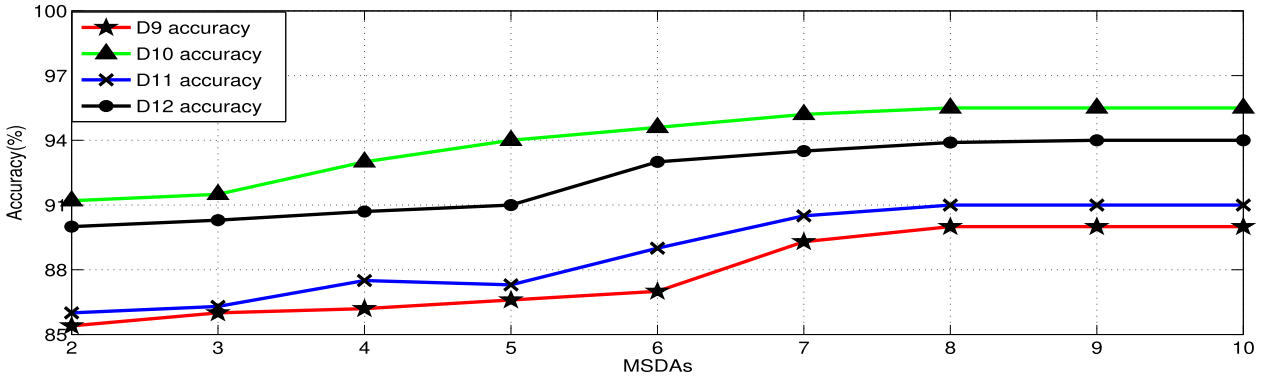


FIGURE 5. Relationship of the number of MSDAs vs the final accuracy on different subsets of UNSW NB15 1 and 2 datasets. The final accuracy saturates beyond using 8 MSDAs.

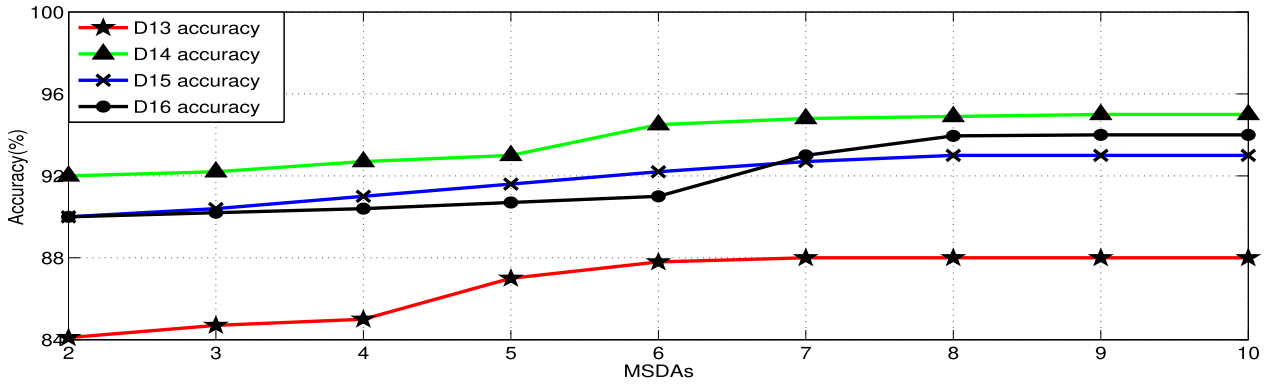


FIGURE 6. Relationship of the number of MSDAs vs the final accuracy on different subsets of UNSW NB15 3 and 4 datasets. The final accuracy saturates beyond using 7 MSDAs.

features from the training data. The LSTM [39] based method achieves better results than the other compared methods. However, due to the fusion of multiple models, our proposed method achieved better results than the standalone LSTM based method.

A. EFFECT OF NUMBER OF MSDAS USED ON ACCURACY

We analyze the accuracy of the proposed method by fusing a different number of MSDAs to build the final classifier. These experiments were performed on the D1-D16 datasets. Specifically, we vary the number of MSDAs to be fused from 2 to 10. The number of layers in each MSDA is randomly chosen from the set $L_m = [1, 3, 5, 7, 9, 11]$. These MSDAS are then fused using the MKLDR algorithm to obtain a final classifier. Figure (3)(4)(5)(6) shows the accuracy of varying the number of MSDAs in case of each dataset. It can be observed that for each dataset the accuracy increases as we increase the number of MSDAs. However, the accuracy

saturates at around 6 MSDAs. This trend shows that the proposed fusion method is effective.

VIII. CONCLUSION

We presented a DDoS attack detection system based on multilevel deep learning technology. The overall system is targeted towards more accurate and more efficient DDoS attack detection in the smart grid network. Our algorithm exploits both shallow and deep auto-encoders for learning powerful features in an unsupervised manner. Features from multilevel auto-encoders are combined using Multiple Kernel Learning (MKL) that automatically learns the weights of the features in the ensemble. Experiments are performed on two benchmark DDoS attack detection databases (and their subsets) and the results are compared with six state-of-the-art methods. Our results show that the proposed method outperforms the compared methods in terms of accuracy and simplicity. In the future, this work can be implemented in a run time environment for securing against DDoS attacks.

REFERENCES

[1] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the Internet of things: A review," in *Proc. Int. Conf. Comput. Sci. Electron. Eng.*, Mar. 2012, pp. 648–651.

[2] S. Mehrdad, S. Mousavian, G. Madraki, and Y. Dvorkin, "Cyber-physical resilience of electrical power systems against malicious attacks: A review," *Current Sustain./Renew. Energy Rep.*, vol. 5, no. 1, pp. 14–22, 2018.

[3] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proc. 29th Int. Conf. Int. Conf. Mach. Learn.*, 2012, pp. 1627–1634.

[4] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, "Multiple kernel learning for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1147–1160, Jun. 2011.

[5] M. Aamir and S. M. A. Zaidi, "Clustering based semi-supervised machine learning for DDoS attack classification," *J. King Saud Univ.-Comput. Inf. Sci.*, 2019. 10.1016/j.jksuci.2019.02.003.

[6] K. Wang, M. Du, S. Maharjan, and Y. Sun, "Strategic honeypot game model for distributed denial of service attacks in the smart grid," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2474–2482, Feb. 2017.

[7] R. C. Diovu and J. T. Agee, "A cloud-based openflow firewall for mitigation against DDoS attacks in smart grid AMI networks," in *Proc. IEEE PES PowerAfrica*, Jun. 2017, pp. 28–33.

[8] P. Srikantha and D. Kundur, "Denial of service attacks and mitigation for stability in cyber-enabled power grid," in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. (ISGT)*, Washington, DC, USA, Feb. 2015, pp. 1–5.

[9] P. Varalakshmi and S. T. Selvi, "Thwarting DDoS attacks in grid using information divergence," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 429–441, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11002093

[10] P. A. R. Kumar and S. Selvakumar, "Distributed denial of service attack detection using an ensemble of neural classifier," *Comput. Commun.*, vol. 34, no. 11, pp. 1328–1341, 2011.

[11] W.-Z. Lu, W.-X. Gu, and S.-Z. Yu, "One-way queuing delay measurement and its application on detecting DDoS attack," *J. Netw. Comput. Appl.*, vol. 32, no. 2, pp. 367–376, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804508000490

[12] J. L. Berral, N. Poggi, J. Alonso, R. Gavaldà, J. Torres, and M. Parashar, "Adaptive distributed mechanism against flooding network attacks based on machine learning," in *Proc. 1st ACM Workshop Workshop AISec*, 2008, pp. 43–50.

[13] X. Xu, Y. Sun, and Z. Huang, "Defending DDoS attacks using hidden Markov models and cooperative reinforcement learning," in *Proc. Pacific–Asia Workshop Intell. Secur. Inform.* Springer, 2007, pp. 196–207.

[14] S. Seufert and D. O'Brien, "Machine learning for automatic defence against distributed denial of service attacks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 1217–1222.

[15] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using SVM and GA," in *Proc. 6th Annu. IEEE SMC Inf. Assurance Workshop*, Jun. 2005, pp. 176–183.

[16] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 76–82, Oct. 2002.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[18] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.

[19] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, "Learning invariant features through topographic filter maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1605–1612.

[20] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1096–1104.

[21] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 1096–1103.

[22] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 833–840.

[23] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[24] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, Jun. 2010, pp. 807–814.

[26] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learn.*, Jul. 2004, p. 6.

[27] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 352–359.

[28] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Jan. 2004.

[29] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 775–782.

[30] S. Sonnenburg and G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, Jul. 2006.

[31] R. Kachouri, K. Djemal, and H. Maaref, "Multiple kernel weighting based SVM for heterogeneous image recognition system," *Int. J. Signal Imag. Syst. Eng.*, vol. 4, no. 2, pp. 60–70, 2011.

[32] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 2126–2136.

[33] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[34] D. Cai, X. He, and J. Han, "Semi-supervised discriminant analysis," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–7.

[35] H.-T. Chen, H.-W. Chang, and T.-L. Liu, "Local discriminant embedding and its variants," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 846–853.

[36] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

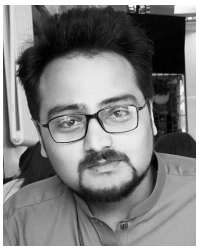
[37] M. Zekri, S. El Kafhali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," in *Proc. 3rd Int. Conf. Cloud Comput. Technol. Appl.*, Oct. 2017, pp. 1–7.

[38] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Secur. Privacy Workshops*, May 2018, pp. 29–35.

[39] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput.*, May 2017, pp. 1–8.

[40] (2012). *UNB ISCX Intrusion Detection Evaluation 2012 DataSet*. Accessed: Jan. 2018. [Online]. Available: http://www.unb.ca/cic/datasets/ids.html/

[41] (2015). *UNSW-NB 15 Dataset*. Accessed: Feb. 2018. [Online]. Available: http://www.ucd.ie/issda/data/commissionforenergyregulationcer/



SHAN ALI was born in Taxila, Pakistan. He received the B.S. and M.S. degrees in electrical (computer) engineering from COMSATS University Islamabad, Wah Campus, in 2016. He is currently pursuing the Ph.D. degree in information security with North China Electric Power University. His research interests include information security, big data analysis, machine learning, and artificial intelligence.

YUANCHENG LI received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2003. From 2004 to 2005, he was a Postdoctoral Research Fellow with the Digital Media Lab, Beihang University, Beijing, China. From 2009 to 2010, he was a Postdoctoral Research Fellow with the Cyber Security Lab, College of Information Science and Technology, Pennsylvania State University, PA, USA. Since 2005, he has been with North China Electric Power University, where he is currently a Professor and the Dean of the Institute of Smart Grid and Information Security.