Auto-Threshold Deep SVDD for Anomaly-based Web Application Firewall

Ali Moradi Vartouni¹, Matin Shokri¹, and Mohammad Teshnehlab²

¹Faculty of Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran

²Faculty of Computer and Electrical Engineering, K.N. Toosi University of Technology, Tehran, Iran

Protecting websites and applications from cyber-threats is vital for any organization. A Web application firewall (WAF) prevents attacks to damaging applications. This provides a web security by filtering and monitoring traffic network to protect against attacks. A WAF solution based on the anomaly detection can identify zero-day attacks. Deep learning is the state-of-the-art method that is widely used to detect attacks in the anomaly-based WAF area. Although deep learning has demonstrated excellent results on anomaly detection tasks in web requests, there is trade-off between false-positive and missed-attack rates which is a key problem in WAF systems. On the other hand, anomaly detection methods suffer adjusting threshold-level to distinguish attack and normal traffic. In this paper, first we proposed a model based on Deep Support Vector Data Description (Deep SVDD), then we compare two feature extraction strategies, one-hot and bigram, on the raw requests. Second to overcome threshold challenges, we introduce a novel end-to-end algorithm Auto-Threshold Deep SVDD (ATDSVDD) to determine an appropriate threshold during the learning process. As a result we compare our model with other deep models on CSIC-2010 and ECML/PKDD-2007 datasets. Results show ATDSVDD on bigram feature data have better performance in terms of accuracy and generalization.

Index Terms—Web Application Firewall, Anomaly Detection, Deep Support Vector Data Description, Convolutional Neural Network, Anomaly Score.

I. INTRODUCTION

The number of attacks on applications, which provide services on the Internet, have dramatically increased. Web application firewall (WAF) is a special type of application firewall applied to applications for protecting, monitoring, analyzing, and filtering HTTP traffics. Websites and web applications use hyper-text transfer protocol (HTTP), which are subject to a number of attacks such as XSS and SQL injection, to deploy internet-based communications. To detect and prevent attacks on those applications WAF plays an important role.

As a solution, WAF applies a set of previously defined rules as signatures, which allows for the detection of vulnerability and malicious HTTP requests. Due to the changing nature of attacks when new attacks were discovered, the signatures must be updated. Due to the changing nature of attacks when new attacks were discovered may have had adverse effects on the applications. Therefore, the signatures must be updated. In preventing this problem, known as zero-day attacks, researches have suggested technical approaches based on anomaly detection that determines malicious patterns that appear in the HTTP request. The foremost problem is that the malicious patterns in data do not conform to a well-defined notion of normal behavior. It means the requests will be rejected, provided that the anomaly score exceeding a predefined threshold. One solution to overcome this problem is anomaly detection, which is able to detect zero-day attacks.

To overcome this problem, several machine learning techniques are presented in anomaly detection for several applications. Usually data patterns do not conform to the definition of common statistical analysis. Therefore, it is necessary that anomaly detection approaches like machine learning methods

appropriately trained in these domains. Recently Neural networks have been used by researchers that show have some key advantages in dealing with high-dimensional data [1], [2].

Several applications of anomaly detection domains such as detecting malicious URLs often have high-dimensional or even ultrahigh-dimensional data [3]. Thus, we must apply feature engineering that can model a suitable representation to preserve the relevant features [4]. Recently, deep learning techniques show that they have impressive performance to capture complex structures in the high dimensional data be applied to anomaly detection [5]. As for HTTP we face a complex hypermedia data, which includes: textual information that is encoded in ASCII format, and can span over multiple lines and be including start line, HTTP header flags, blank line and HTTP body. Thus for analyzing non-stationary and non-structured data we have to use text mining and natural language processing (NLP) techniques to extract features, but these data can be too high dimensional to be learned by traditional machine learning method[6]. So we encounter a deep model as a powerful feature extractor to relief this problem.

Briefly, we introduce the main contributions of the proposed algorithm in this paper under the following consecutive tasks:

- 1) Using end-to-end Deep support vector data description (Deep SVDD) to obtain high-level normal feature vector. This approach is a deep anomaly detection inspired by kernel-based one-class classification and minimum volume estimation [7].
- 2) Obtaining threshold during Deep SVDD learning process. For this purpose we introduce a novel cost function that gives an advantage of discriminating anomalies/normal points.
- 3) Creating a feature set using n -gram and one-hot methods. In addition, Extracting an one-hot feature set using

convolution neural network (CNN) from HTTP requests.

The remainder of the paper is organized as follows. In Section II, we briefly describe several related works including one-class classifier for anomaly detection algorithms. Then we review deep learning models for anomaly detection, and we highlight related works in WAF based machine learning, especially deep neural networks. The proposed method are presented in Section III. This is followed by Section IV showing the evaluation and experimental results. Then, the discussion of comparing different methods is presented in Section V. Finally, Section VI draws conclusions and suggests future work.

II. RELATED WORKS

Many articles have reviewed anomaly detection in several applications [2]. Also, the survey [5] studied deep anomaly detection techniques and particularly intrusion detection systems [8]. However, we first briefly discuss one class classifications that play major role in anomaly detection predictions. Consequently, considering the importance of deep learning in artificial intelligence fields, we review works on neural network and deep anomaly detection. Then, we present solutions for WAF based on machine learning and especially deep learning.

A. One-class classifiers

1) Reconstruction error (RE)

Auto-encoder (AE) is an unsupervised neural network, whose objective is to learn to reproduce input vector x as output $\hat{x}$ such to minimize the reconstruction error, which is defined in (1) for a D dimensions entry [9].

$$RE = \sqrt{\sum_{i=1}^D (x_i - \hat{x}_i)^2} \quad (1)$$

One of the important applications of the AE is anomaly detection [10]. AE represents features in new space that estimate regular data features in the small number of neurons in a hidden layer. As the AE parameters are fitted and adjusted based on the normal data, these have lower reconstruction errors than abnormal instances. Therefore, the higher the reconstruction error, the higher the possibility of that data point being an anomaly.

2) Density Estimation

Density-based techniques perform anomaly detection by estimating the density of training data. One of the density estimation approaches that can be used for constructing an anomaly detection model is kernel density estimation (KDE) [11]. In statistics, KDE is a non-parametric way to estimate the probability density function of a random variable. Let $\{x_1, x_2, \dots, x_n\}$ be an independent and identically distributed sample drawn from some distribution with an unknown density f . We are interested in estimating the shape of this function f . Its kernel density estimator is according to (2).

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^n K_h\left(\frac{x - x_i}{h}\right) \quad (2)$$

where K is the kernel — a non-negative function — and $h > 0$ is a smoothing parameter called the bandwidth. A kernel with subscript h is called the scaled kernel and defined as $K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right)$.

3) one-class support vector machine (OCSVM)

The method of SVDD takes a spherical, and not a planar, approach [12]. Another approach of OCSVM classifier separates all the data points from the input feature space and maximizes the distance from this hyperplane to this space according to quadratic programming minimization functions as in (3).

$$\begin{aligned} \min_{\omega, \varepsilon_i, \rho} & \frac{1}{2} \|\omega\|^2 + \frac{1}{\vartheta n} \sum_{i=1}^n \varepsilon_i - \rho \\ \text{s.t.} & (\omega \cdot \varphi(x_i)) \geq \rho - \varepsilon_i \quad \text{for } i = 1, \dots, n \\ & \varepsilon_i \geq 0 \quad \text{for } i = 1, \dots, n \end{aligned} \quad (3)$$

(ω, ρ) is a weight vector and an offset parametrizing a hyperplane in the feature space associated with the kernel.

4) Ensemble isolation forest (IF)

This method partitions data using a random tree until all instances are isolated from each other. Firstly, the algorithm builds isolation trees using sub-samples of the training set and, later, it calculates the isolated path length of the observation through isolation trees to obtain the anomaly score for each instance. The anomaly score is defined as (4).

$$s(x, n) = 2 - \frac{E(h(x))}{c(n)} \quad (4)$$

where $h(x)$ is the path length of instance x , $c(n)$ is the average path length of unsuccessful search in the tree, the $E(h(x))$ is the average path length of every instance across different trees, and n is the number of data points in a chosen sub-sample. To building a tree, the algorithm selects a feature randomly; and it selects a split value in the values range of the selected feature in order to isolate observations [13].

5) Elliptic envelope

This method tries to find the shape of the data and mark data as anomalous that are far from the shape. Thus, it fits the smallest volume ellipsoid using a robust covariance estimation to the data by modifying the free parameter contamination [14]. The main task of the covariance elliptic envelope is finding observations whose covariance matrix has the lowest determinant.

B. Neural network and deep anomaly detection algorithms

a) Hybrid deep neural networks(DNNs) and one-class classifier: Since the problem, known as the ‘curse of dimensionality’ is an obstacle for many anomaly detection methods, this technique uses DNNs as feature extractors. The compressed and extracted features from the hidden representations of the auto-encoders or restricted Boltzmann machines (RBMs) are input to traditional anomaly detection algorithms to detect outliers [15]. For example, according to Figure 1(a) auto-encoders reduced features and then, one-class classifiers such as one-class support vector machine (OC-SVM) [16], kernel density estimation (KDE) [17] detected outliers, or RBMs applied to extract features and then, OCSVM detected anomalies [16].

b) Self-representation: As we mentioned, for anomaly detection, the AE are trained on the normal data [18]. AEs play a fundamental role in unsupervised learning and deep architectures for transfer learning and other tasks. Figure 1(b) shows that the reconstruction error must be compared by a threshold value in a test phase. When layers become deeper, the network can be able to extract more complex features than traditional shallow networks. In stack auto-encoder (SAE) techniques, where each layer trains one after another, the structure of SAEs is stacking AEs into hidden layers by an unsupervised layer-wise learning algorithm.

c) Variational auto-encoder (VAE): VAE is one of the generative machine learning techniques, which deals with models of distributions of data points in some potentially high-dimensional space [19]. The objective function of a VAE is the variational lower bound of the marginal likelihood of data since the marginal likelihood is intractable. The reconstruction probability from the VAE can be considered as an anomaly score [20]. A shrink regularization added to the loss function of auto-encoder, which is designed to penalize normal data points whose vectors in the latent space are a large magnitude that is it will restrict the normal data to lie close to the origin [21]. Then VAE attempts to encode data that are distributed as a standard Gaussian in the latent space (Figure 1(c)).

d) Adversarial learned one-class classifier: In this strategy, During inference, the trained model is expected to accept in-class examples and reject out-of-class examples while, in other scenarios, the trained model try to accept only in-class examples. Therefore, the model trained adversarially with the ability to encode and reconstruct instances according to Figure 1(d). One section i.e. generator outputs random samples from random latent vector and another i.e. discriminator that classifies normal and outlier samples [22], [23], [24].

e) SVDD and Deep SVDD: This method obtains a spherically shaped boundary around normal data sets. The aim is to minimize the volume of the hyper-sphere by minimizing squaring of the radius and with the constraint that the sphere contains all training samples [25]. Furthermore, the Deep SVDD learns a neural network transformation that attempts to map most of the data network representations into a hyper-sphere characterized SVDD conditions (Figure 1(e)) [7]. In other study, the one class neural network (OC-NN) is a neural architecture using a loss function based on OCSVM [26].

C. WAF based on anomaly detection

The first study of intrusion detection based on web application security introduced by Kruegel and Vigna [27]. They analyzed statistical characteristics of query and parameters of HTTP requests. Also, they extend their works in the multi-model approach in 2005 [28]. Many types of research apply machine learning to investigate this issue. Thus, in this section we introduce a number of related studies.

a) WAF based on traditional machine learning: The affinity propagation (AP) as an unsupervised method is used on real HTTP traffic streams [29], [30]. They used the character distribution of each source in HTTP requests as the attributes. Due to the importance of minimizing false

positives, probabilistic automaton such as Markov models has been used in many studies to detect web attacks [31], [32], [33]. First, HTTP requests are tokenized, then those tokens are fed to the deterministic finite automata (DFA) algorithm [31]. While papers [32] and [33] use Markov chain model based on other features constructing from n-gram method. Furthermore, the extreme learning machine (ELM) as one of the neural network models [34], and binomial logistic regression [35] classified normal and anomaly web traffic on tokenized retrieved requests. In addition, probability distributed model, Hidden Markov Model (HMM), and one-class support vector machine (OCSVM) model detected web attacks on attribute values [36].

b) WAF based on deep learning: Recurrent neural network (RNN), SAE, DBN, and CNN are various categories of deep neural networks that are used for web application firewalls. In [37] character-level CNN are used for web attacks detection. They used supervised learning for anomaly detection that classified training data by normal data and attacks. Many researchers have applied SAE to implement WAF in [38], [39], [40] and denoising SAE in [41] for web attacks detection that is applicable for extraction and dimensional reduction of features vectors. In addition to using SAE, we used DBN model for feature extraction in [16]. We then used three one-class classifiers called OC-SVM, ensemble isolation forest, and covariance elliptic envelope for the detection of attacks. Then they compared different methods with each other. The deep-based RNN includes Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) used for WAF in [42], which can be efficient for sequential and time-dependent data such as HTTP requests. This model takes uniform resource locators (URLs) in the HTTP GET requests as the input. Subsequently, the URLs are tokenized by following a particular strategy: First, two RNNs learn the normal request patterns and outputs their familiarities with given URLs. Then a trained neural network decides whether given requests are anomalous or not based on the output of the RNNs. Also, we proposed an auto-encoder LSTM (AE-LSTM) architecture for feature extraction HTTP data to detect attacks in WAF [43]. Instead of fully connected layers in SAE, the LSTM layers can extract sequential data. Thus, the one-hot model of HTTP is also used in the AE-LSTM model. In addition, ensemble isolation forest is used on last extracted features to detect attacks. Also, C-LSTM consists of CNN and LSTM layers are presented to model spatiotemporal data contained in traffic data effectively as one-dimensional time series signal [44].

The one-class classification algorithms are widely used for anomaly detection tasks. For this ability, one-class classifier is used to identify unknown attacks in WAF. This is the key reason that we use anomaly-based techniques for WAFs. In addition, the HTTP data is semi-structure data with non-stationary sequences. Therefore, feature engineering techniques like NLP approaches and unlike HTTP tokenizer, is necessary to consider data generality. Moreover, as recent researches on WAF based on anomaly detection have worked on layer by layer feature extraction, the proposed algorithm uses an end-to-end technique for extracting features.

Fig. 1. Anomaly detection based on neural network schemes

III. METHODOLOGY

This paper proposes an Auto-Threshold Deep SVDD (ATDSVDD) to learn HTTP request data from websites and web applications have been received from WAF servers.

In the proposed method we used Deep SVDD model to learn normal requests in the training phase, then the trained model predicts anomalous requests, which have never been seen in the training step. The anomaly scores, which is derived from the value of cost functions can be used to determine anomalous when attacks occur within communications. Additionally, we modify the cost function to generate a threshold automatically to define boundary of normal and attack requests.

An abstract view of the main steps of the proposed model includes feature engineering, one-class classifier, and auto threshold generation are shown in Figures 2 and 3.

A. Feature engineering

The representation of data can have an impressive effect on machine learning models. This feature engineering includes construction and extraction efficient of information from the data. In term of the data type, the HTTP data can be considered text as a sequence of characters, words, and so forth. Therefore, the textual data must be mapped to real valued vectors and transform tokens into features.

One of the simplest techniques to numerically represent text is in counting of occurrence the particular token in a HTTP text. However, this scenario loses item order. The items can be character, word and etc. For this, discrete text mining techniques such as n -gram and one-hot methods that consider the order of items are utilized.

1) Bigram

In the first scenario, bag of words can be used. In this method items are counted separately throughout the text. Items depend on counting strategy, for example we can count

Fig. 2. ATDSVDD-bigram: The scheme of proposed anomaly detection method for WAF based on bigram

Fig. 3. ATDSVDD-onehot: The scheme of proposed anomaly detection method for WAF based on one-hot

alphabets or words. In this scenario because the nature of the sequential text structure may not have valuable information on counted items. $n > 2$ -gram is the strategy to count items considering sequence nature of texts. The parameter n simply refers to the number of items and the n -gram used to capture statistical information from data set, which produces lists of sequences of items. On the other hand, the bigram model i.e. $n = 2$ in n -gram, approximates the probability of token given all the previous tokens by using only the conditional probability of one preceding token. Then, when we use a bigram model to predict the conditional probability of the next item, the probability of an item depends only on the previous item, which is also known as Markov assumption [45]. Furthermore, we can further generalize the bigram model to the trigram model ($n = 3$) and so forth. As n increases, the amount of data required for modelling is increased. In addition, the main drawback of this method is the growing size of feature space exponentially as growing amount of n . Choosing a large amount of n leads to ill-posed problems.

Using of two benchmark corpora i.e. n -gram character-level

and word-level strategies, provides evidence that character-level information has better performance to discriminate between spam and legitimate languages [46]. The most important property of the character-level n -gram approach is that it avoids using of language-dependent tools such as tokenizer and also reduces the need of a lot of storage space. For this purpose we define a matrix v with $m \times n$ dimensions $m, n \in \mathbb{N}$. v^{ij} is equal to the frequency of occurrences of the j^{th} bigram item in the i^{th} request where $i \in \{1, 2, \dots, m\}$ and $j \in \{1, 2, \dots, n\}$ respectively.

2) One-hot

A one-hot encoding is a representation of categorical variables. In natural language processing, a one-hot vector based on character is a $1 \times k$ vector is used to distinguish each character by its alphabet. All elements of vector are 0s except for one of the cells that indicates a special character. For this scenario, we can stack these one-hot encoding to preserve ordering of items. Due to the high cardinality-categorical variable problem, we consider the common model called character-level ConvNet [47] instead of word-level one-hot

representation. Thus, in the present work, we train a convolutional neural network (CNN) with multi layers of convolution on top of one-hot vectors. Addition to parameter sharing, CNNs utilize layers with convolving filters that are applied to local features[48].

B. Deep SVDD as one-class classifier

[7] introduced a novel anomaly detection method (Deep SVDD), which is trained on normal examples based on minimizing distances of features map to a center c . Minimizing the volume of the hyper-sphere forces the network to extract features as the network must closely map the data points to the center of the sphere. Deep SVDD utilize end to end unsupervised approach, which maintains a low intra-class variances in the feature space for the normal class.

The Deep SVDD objection function is as follows:

$$\min_W \frac{1}{n} \sum_{i=1}^n \|\phi(x_i; W) - c\|^2 \quad (5)$$

Where $F \subseteq R^d$ and $X \subseteq R^p$ are input and output space respectively, and $c \in F$ is the center of output space. Also $W = \{W_1, W_2, \dots, W_l\}$ is considered as the weight of the neural network with $l \in \mathbf{N}$ layer. Considering ϕ as a network, let $\phi(\cdot; W) \in F$ is output of neural network, the objective function tries to minimize the distance of $\phi(\cdot; W)$ and center c , which leads to enforce the network to extract features, which is mapped to a vicinity of c as finding a data-enclosing hyper-sphere of smallest size. Hyper parameter c can be initialize by average of $\phi(x; W)$ where inputs space x are normal training data.

For a given test point $x \in X$, an anomaly score function $R(x)$ can be defined as follows:

$$R(x) = \|\phi(x; W^*) - c\|^2 \quad (6)$$

Where W^* are the network parameters of a trained model. As the discrimination of anomalous and normal data must be able to define a threshold R , then we can define a function to discriminate attack from non-attack requests. The function can be demonstration as bellow:

$$f(x) = \begin{cases} x \text{ is normal} & : R(x) < R \\ x \text{ is anomalous} & : R(x) \geq R \end{cases} \quad (7)$$

Whereas, normal examples of the data are closely mapped to center c , and anomalous examples are mapped further away from the center. Thus, R determines hyper-sphere boundary of normal and abnormal data.

C. Threshold

c and R are main characteristics that determine the locus of any hyper-sphere, Although c can be initiated by calculating the mean of the network outputs $\phi(x, W)$ [7], attaining appropriate R as a threshold is a challenging problem.

As mentioned in III-B to predict on a trained network over test data we have to calculate a threshold from training data, but because we do not have information about scattering

distribution of output ($\phi(\cdot, W)$) in the hyper-sphere, there is no standard manner to calculate an appropriate R (R^*). For example we can assume that R^* is an average of all scores on training data or in another scenario R^* can be maximum score. This threshold does not prepare a confidence for the perspective of suitable results before being tested.

To deal with the threshold problem, we try to the change scattering distribution of data in the hypersphere. For this purpose we present a novel cost function to force the network to extract features to prepare both minimization distance of $\phi(\cdot; W)$ and c and keeping the data scattering distribution, in the Gaussian distribution with center c and standard deviation σ . σ also is updated in every step on optimization Fig.[2, 3]. This enforce mapped features to drop in σ area around of c . Therefore, this property enables us to define a reliable threshold by statistically analyzing output cost. According to this case we can propose the threshold $R^* = c + 3\sigma$ to assure that 99.7% of the training data with the least scatter around the c , fall inside of R^* aria. As a result, in this case we have two parameter c and σ leads to obtaining R^* , whereas in euclidean distance measure we have only c and hence cannot calculate appropriate R .

According to the mentioned strategies we present the cost function as follows:

$$\min_W 1 - \frac{1}{n} \sum_{i=1}^n e^{-\frac{\|\phi(x_i; W) - c\|^2}{2\sigma^2}} \quad (8)$$

Where σ change in every optimization step. This cost function tries to minimize hyper-sphere considering by mapping the feature space to the Gaussian space. The optimization pseudo-code demonstrated as follow:

Algorithm 1: ATDSVDD Optimization algorithm

input : σ_0, c, W_0 , number of epochs = $epochs$, batch size = m , size of training data = n , $e = 1$, $i = 0$

output: σ^*, W^*

Initialize $\alpha \leftarrow \frac{m}{n}$

while $e \leq epochs$ **do**

while $(i + 1)m + i \leq n$ **do**

$x_i \leftarrow i^{th}$ batch of training data

 Apply x_i on W_i :

 • $W_i \leftarrow \arg \min_{W_{i-1}} 1 - \frac{1}{n} \sum_{j=1}^n e^{-\frac{\|\phi(x_j; W_{i-1}) - c\|^2}{2\sigma_{i-1}^2}}$

$o_i \leftarrow \phi(x; W_i)$

$\sigma_{o_i} \leftarrow$ compute standard deviation of o_i

$\sigma_i \leftarrow \alpha \sigma_i + (1 - \alpha) \sigma_{o_i}$

$i \leftarrow i + 1$

end

$e \leftarrow e + 1$

$\alpha \leftarrow \frac{m}{en}$

end

Optimization algorithm tries to maximize similarity between x and c considering current σ in each step. After that we update σ according to the new feature space that was optimized in the last step and that leads to the decrease of σ . This is

repeated until the minimum σ that the feature space can fall into is determined.

Another challenge that we tackle is the anomaly score problem. Due to the nature of Euclidean distances of defining set of numbers that are not bounded, anomaly scores can not be defined properly. Basically, we do not know from its size whether a coefficient indicates a small or large distance. According to cost function (8) that is bounded between 0 and 1, whereas, for the Euclidean distance we have not an obvious bound for maximum distance, we can define a suitable anomaly score function that can determine the probability of the anomaly. For this purpose the anomaly score can be defined as:

$$R(x) = 1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}} \quad (9)$$

One of the problems in threshold handling is determining a threshold to ensure that whether it is suitable for our application or not. Choosing an appropriate threshold depends on the application that we can handle by the coefficient of σ parameter in our model. In Gaussian distribution, σ shows how distribution can stretch/scale. About %68 of normal values, drawn from a normal distribution, are within 1σ away from the mean. About %95 of the normal values are within 2σ , and about %99.7 of those are within 3σ that can be shown by $\alpha\sigma$. As we increase our α we can expect to decrease TPR and FPR where FPR is more desirable for our application in a reasonable TPR. Conversely, when we decrease our alpha we expect to increase TPR and FPR.

D. Proof

Deep SVDD [7] claims that a hypersphere with optimum radius R^* exists in which all of the points fall as much as possible. The important point is that the optimization algorithm does not consider any distribution for data in the hyper-sphere, hence after training, we have to apply some statistical computation such as max, min, or quantile for obtaining a threshold to use it in the test phase. If there is a skew in the distribution of points, we have difficulty obtaining threshold (for example Figure 4(a)). For tackling this problem, first we change the optimization equation $\|\phi(x;W) - c\|^2$ to $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}}$ to force model to extract features that distribution of points in the hypersphere follows Gaussian distribution (such as Figure 4(b)). For a constant σ , we know $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}}$ will be minimized when $\|\phi(x;W) - c\|^2$ is minimized like deep SVDD which is optimal.

Second, we consider sigma as a control parameter to obtain suitable threshold. for this purpose we change $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}}$ to $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}}$ $\sigma \rightarrow \sigma^*$, then for obtaining σ^* we gradually update it into minimum σ where points can fall down into. For this purpose we design a iterative algorithm that first, we consider σ as a constant value and minimize $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma^2}}$ then update σ into σ^* using iterative stochastic algorithm [49] demonstrated in Figure 5, as follows:

- 1) Minimize $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma_i^2}}$ whit σ_i which at the first is a constant value σ_0

- 2) Estimate σ_{i+1} which is created by i th data batch that is applied on $1 - e^{-\frac{\|\phi(x;W)-c\|^2}{2\sigma_i^2}}$
- 3) Update σ_i using proportion of σ_{i+1} by equation as follows:
 - $\sigma_i \leftarrow \alpha_i \sigma_i + (1 - \alpha_i) \sigma_{i+1}$: α_i is $(\frac{m}{in})$ that m is batch size and n is number of data
- 4) If $\sigma_i < \epsilon$ (ϵ is a small number), go to line 1, else terminate
 - If following condition are met, we can expect that σ converges to σ^* :
 - 1) $\sum_i \alpha_i = \infty$. For this condition we can show $\sum_i \frac{m}{in} = \frac{m}{n} \sum_i (1 + \frac{1}{2} + \frac{1}{3} + \dots) = \infty$
 - 2) $\sum_i \alpha_i^2 < \infty$. For this condition we have $\sum_i (\frac{m}{in})^2 = (\frac{m}{n})^2 \sum_i (1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots) < \infty$

IV. EXPERIMENTS

To evaluate the effectiveness of the proposed framework, we compare it with the state-of-the-art methods for WAF systems based on anomaly detection criteria. Since stacked auto-encoders and one-class classifier can be used for feature representation and anomaly detection respectively, in all studied methods, we first summarize other WAF systems based on anomaly detection and then we present the datasets, evaluation criteria, and experimental results.

A. Datasets

We evaluate our experiments with two public datasets, CSIC 2010 dataset [50] and ECML/ PKDD 2007 dataset [51]. The datasets contain only HTTP traffics in order to train the anomaly detection models by normal traffic set.

The training sets of CSIC and ECML contain 36000 and 20000 normal data, respectively, and test sets contain 36000 and more than 15000 normal data also, more than 25000 and 15000 abnormal data, respectively.

The HTTP requests, in addition normal to traffics include several attacks such as SQL injection, cross-site scripting (XSS), buffer overflow, files disclosure, XPATH injection, LDAP injection, information gathering, parameter tempering and so on.

B. Evaluation criteria

The fit generalization is necessary for anomaly detection especially WAF systems. The generalization is a trade-off between false positive and false negative rates [31]. Therefore,

Fig. 4. Figures show the distribution of output data according to different cost functions in two-dimensional space (a) Euclidean cost function and (b) Gaussian cost function. The final projection of data points in (a) has a skew that causes having difficulty obtaining a threshold in a spherical manner. This threshold either doesn't cover all normal points in its space -when we choose a small threshold- or not to be able to fit those in its space when we choose a larger threshold. Conversely, In the Gaussian cost function, the final projection of data will be mapped at the Gaussian space -because it forces the model to extract features fitting to this cost function- therefore it allows us to choose threshold appropriately.

Fig. 5. This figure shows updating steps in iterative stochastic algorithm

TABLE I
CONFIGURATION OF THE ONE-HOT MODEL

layer	# filters, # neurons	kernel size	stride	output feature size
Embedded	32	-	-	32×2500
CNN 1	128	7	1	padding=Valid
CNN 2	256	7	2	padding=Valid
CNN 3	256	5	2	padding=Valid
CNN 4	128	3	2	padding=Valid
CNN 5	32	3	2	512
FC 1	512	-	-	1024
FC 2	1024	-	-	1024
FC 3	1024	-	-	100

in addition to accuracy, we also use several measures to evaluate the performance of methods according to:

$$\begin{aligned}
 \text{accuracy}(Acc) &= \frac{\text{number of true detected}}{\text{number of all instances}(N)} \\
 \text{recall}(DR) &= \frac{\text{number of relevant attacks detected}}{\text{number of relevant attacks}} \\
 \text{precision}(PR) &= \frac{\text{number of relevant attacks detected}}{\text{number of attacks detected}} \\
 \text{specificity}(Spec) &= \frac{\text{number of relevant normal detected}}{\text{number of relevant normals}} \\
 f1\text{-measure} &= \frac{2 \times DR \times PR}{DR + PR}
 \end{aligned} \tag{10}$$

Also, we use ROC curves that provide insights closer to false positive rates and true positive rates for different thresholds of scores.

C. Implementation setup

Experiments were carried out on a system with the following specifications: X86-64 GNU/ Linux, Intel® Core™ i7-7700k CPU @4.20 GHz, GeForce GTX 1080 Ti GPU, and 47 GB RAM. Additionally, algorithms were implemented in Python 3.6 programming language with Tensorflow and Keras libraries. These libraries are useful tools to design, build, and train deep learning models, which have been developed by Google [52], [53].

As mentioned in sections III-A1, III-A2 we choose two input features (bigram and one-hot) for network training. For this purpose we use 96 ASCII codes from 256. They are sufficient for examining individual HTTP requests [54]. Therefore, the bigram feature vector is represented by $96 \times 96 = 9216$ dimensions. Also, the one-hot vector dimension is 96 multiplied by the length of request (total characters) in individual HTTP request. The maximum length that we consider is 2500.

As for the bigram and one-hot feature space we use two types of architecture. For one-hot feature we use the model character-level ConvNet model [47] described in table I and bigram feature space uses the deep MLP model with 4000, 1000, 400, and 100 neurons for each layer respectively. In addition, the activation functions for our architectures used in each layer are LeakyReLU [55]. Furthermore, the batch size for all methods is 64 and the number of epochs is 30.

TABLE II
THE RESULTS OF PROPOSED METHOD BASED ON BIGRAM AND ONE-HOT METHODS)

Data	Method	Acc.	DR	PR	Spec.	F1
CSIC	bigram	90.63	92.43	85.83	89.37	89
	one-hot	81.7	85.98	73.89	78.85	79.48
ECML	bigram	80.64	78.77	81.95	82.53	80.33
	one-hot	60.97	63.93	60.51	57.99	62.18

Fig. 6. The ROC curves of CSIC dataset for bigram and one-hot methods

D. Analysis of feature construction methods

In this section evaluation are analyzed for each feature method (bigram, one-hot) and then compare them with each other on CSIC and ECML datasets. Also, we consider the Deep SVDD learning rates 10^{-7} and 10^{-5} for bigram and one-hot feature models, respectively.

Table II reveals bigram method has better performance than one-hot considering criteria evaluation that mentioned in section IV-B. Fig. 6 and 7 shows ROC curve for each method on CSIC and ECML respectively. It also can show that the bigram method obtains higher value than the one-hot method. Thus, our preferred method, which is chosen for further investigation is the bigram method.

E. Analysis of methods comparison

1) Methods compared

In all approaches, we use bigram method to construct features from HTTP requests. In addition, since the bigram featuring size is large, according to the following approaches, we use the stack auto-encoder (SAE) to extract efficient features for anomaly detection.

a) **SAE-RE**: In this method, the reconstruction error of AE is calculated [40]. The anomaly score is the reconstruction error of the last layer that is the accumulated of all errors. The

Fig. 7. The ROC curves of ECML dataset for bigram and one-hot methods

Fig. 8. The ROC curves of CSIC dataset for every method

appropriate threshold is essential, which triggers the trade-off between true positive rate (TPR) and false positive rate (FPR). The paper [40] computed the threshold based on the average and standard deviation reconstruction error.

b) **SAE-KDE**: In this method, the density estimation is used on the last hidden layer of SAE [17]. The Gaussian kernel with bandwidth equal to $\sqrt{\frac{\text{hidden-size}}{2}}$ is used for KDE. In this method, like the previous method, the appropriate threshold is a crucial parameter.

c) **SAE-OCSVM**: The OCSVM estimates the support of distribution on the last layer of SAE [16]. This method requires the choice of a kernel and a scalar parameter to define a frontier. The parameter ϑ , also known as the margin of the OCSVM, corresponds to the maximum proportion of outliers in the training data and this value is set by the user itself.

d) **SAE-IF**: The IF measures normality based on decision function on the last layer features of SAE [38], [16].

e) **SAE-Elliptic**: The elliptic envelope estimates the Gaussian distribution on the last layer data of SAE [16].

2) Experimental results

Results of our models against other methods are listed in Table III for CSIC and ECML data. Also, we can compare ROC curves of different methods in Figures 8 and 9 for CSIC and ECML datasets, respectively. To have the same conditions, the dimension of all methods reduces to 100 features and the learning rate for AEs and SAEs is adaptive in each layer [56].

F. Analysis of threshold strategy

In this section we compare the proposed exponential cost function with Euclidean cost function for SVDD classifier on WAF datasets. As we mentioned, there is no information about scattering distribution of output in the hyper-sphere for

Fig. 9. The ROC curves of ECML dataset for every method

TABLE III
EVALUATION OF CSIC DATA WITH VARIOUS METHODS

Data	Method	Acc.	DR	PR	Spec.	F1	AUC
CSIC	Proposed	90.63	92.43	85.83	89.37	89	.97
	SAE-OCSVM	54.6	32.35	42.96	70.09	36.90	.51
	SAE-IF	67.52	57.96	60.97	74.17	59.43	.66
	SAE-elliptic	62.82	52.44	54.93	70.04	53.65	.61
	SAE-KDE	61.31	48.81	53.12	70	50.87	.63
	SAE-RE	75.88	61.31	75.33	86.02	67.60	.85
ECML	Proposed	80.64	78.77	81.95	82.53	80.33	.88
	SAE-OCSVM	59.42	50.15	61.78	68.75	55.36	.59
	SAE-IF	62.15	26.46	93.35	98.1	41.23	.62
	SAE-elliptic	81.59	66.87	95.09	96.53	78.45	.82
	SAE-KDE	62.56	50.01	67.01	75.21	57.28	.66
	SAE-RE	78.63	84.12	75.9	73.1	79.8	.88

TABLE IV
THE COMPARISON OF PROPOSED THRESHOLD VS. EUCLIDEAN METHODS

Data	Method	Acc.	DR	PR	Spec.	F1
CSIC	R_{exp}	90.63	92.43	85.83	89.37	89
	R_{mean}	86.27	99.05	85.3	77.38	85.55
	R_{max}	90.57	83.41	92.91	95.57	87.91
	$R_{0.997}$	90.02	94.14	83.62	87.16	88.57
ECML	R_{exp}	80.64	78.77	81.95	82.53	80.33
	R_{mean}	50.18	100	50.18	0.0	66.82
	R_{max}	68.71	37.96	99.15	99.67	54.9
	$R_{0.997}$	50.18	100	50.18	0.02	66.83

the Euclidean cost function to calculate a threshold. Thus, we roughly assume thresholds based on average, maximum or the quantile of anomaly scores. In obtaining a reliable threshold we design the exponential cost function, which tries to force the output data distribution toward a Gaussian distribution with c and σ parameters. Thus, a threshold can be estimated based on linear composition of average and standard deviation of cost function. Table IV tabulates the comparison of exponential cost function performance with Euclidean cost function performance based on applying various types of threshold. For this purpose, we consider $R_{exp} = c + 3\sigma$ as a threshold for the proposed cost function and also we define different types of thresholds for Euclidean cost function, which are $R_{mean} = \text{mean}\{\text{scores}\}$, $R_{max} = \text{max}\{\text{scores}\}$, and $R_q = q - \text{quantile}\{\text{scores}\}$. The q value in R_q is the number of equal-sized partitions of a finite set. We consider q value to be 0.997 in comparison to R_{exp} value, which both methods use 99.7 percent of normal data for calculating a threshold.

V. DISCUSSION

In this section, we interpret the experiments and explore the significance by focusing on the generalization of different methods on CSIC and ECML datasets.

For constructing features from HTTP data for attack detection, we use bigram and one-hot techniques that can model token's sequences of data by ordering of characters in sequence of requests. In fact, the difference between bigram and one-hot technique is that one-hot model is dependent on length of request. A specific deep model based on convolution neural network used to extract rich features from one-hot features. On the other hand, deep-based approaches are practically one of the best options for large and high dimensional data.

As for generalization, it is a crucial metrics for anomaly detection methods such as intrusion detection systems [31]. To optimize the impact of benign attacks in normal data, we need to acquire appropriate generalization. Figure 10 (a) shows a desirable trade-off between false positive and false negative rates. For having desirable generalization, the threshold value plays a important key role, which it can lead to proper boundary to distinguish normal and anomalous data. Since, there are no sense about the output distribution of data that spread in a hyper-sphere, we can not determine a threshold toward a proper trade off between false positive and false negative rates. For example if that distribution is estimated in a skew and torn shape 10 (b), the threshold can not estimate a fit model boundary.

To compare to other models, we contrast the ATDSVDD with stack auto-encoder and one-class classifiers based on accuracy and generalization. For this purpose we examine F1, which is related to precision and recall (detection rate), and AUC can show the performance of model based on sensitivity (detection rate) measure against false positive (1 - specificity) rate measure. Figure 11 compares the generalization of methods with each other by measuring F1 and AUC criteria. It also examines all models based on accuracy measure.

As we see, F1 and AUC show that the proposed model outperforms one for all datasets, as well as depicting that the gain accuracy is higher than 80% compared to other models. Also this reveals that SAE-RE achieved significant results for all data, while SAE-elliptic gained appropriate results only on ECML data.

As a result, the models based on ATDSVDD not only archived significant results based on generalization and accuracy measure, it can perform as an end-to-end feature extractor unlike SAE layer by layer feature extraction.

VI. CONCLUSION

In this paper, we presented a comparison of unsupervised deep neural network based on Deep SVDD and SAE-OC in the WAF framework. The proposed model was examined using two benchmark corpora CSIC 2010 and ECML/ PKDD 2007 and several evaluation measures that provided a comparison between different methods for detecting anomalies.

The results show that ATDSVDD produced higher performance in respect to accuracy and generalization measures. In this paper, we use character-level bigram and one-hot methods to construct the feature vector. Also, we perform CNN as a feature extractor from the one-hot method. Then, for the learning feature space based on normal data, Deep SVDD classification extracts appropriate represented features. Finally, according to exponential normal distribution, we proposed a novel cost function that maps feature space to the Gaussian space. Therefore, we can consider a suitable threshold around normal data by setting the linear composition of average and standard deviation of anomaly score in HTTP requests.

To work on this issue, we deal with two challenges: feature engineering and anomaly detection. For future works, we can define a feature extraction to extract the semantic feature in HTTP request domain, or perform other methods especially,

Fig. 10. Generalization strategies for all possible HTTP requests, Normal (green region) illustrates the set of normal requests, cross points indicate different attacks. The yellow regions illustrate the estimated model for anomaly detection. (a) is the desired system, (b) is the over-generalized model, and (c) is the under-generalized model.

Fig. 11. Comparison the accuracy and generalization of methods

GAN and VAE models for detection of HTTP attacks. Also to deal with data stream problems incremental learning can be considered. Nowadays adversarial attacks on the model is a great concern in the AI-based security field. Therefore, model robustness to this type of attack should be considered for future work.

REFERENCES

- [1] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.
- [2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: an application of large-scale online learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 681–688.
- [4] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2041–2050.
- [5] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.
- [6] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, 2018.
- [7] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International Conference on Machine Learning*, 2018, pp. 4393–4402.
- [8] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.
- [9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [10] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *CVPR 2011*. IEEE, 2011, pp. 3449–3456.
- [11] M. Nicolau, J. McDermott *et al.*, "One-class classification for anomaly detection with kernel density estimation and genetic programming," in *European Conference on Genetic Programming*. Springer, 2016, pp. 3–18.
- [12] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.
- [14] P. J. Rousseeuw and K. V. Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.
- [15] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, pp. 1–13, 2017.
- [16] A. M. Vartouni, M. Teshnehlab, and S. S. Kashi, "Leveraging deep neural networks for anomaly-based web application firewall," *IET Information Security*, vol. 13, no. 4, pp. 352–361, 2019.
- [17] M. Nicolau, J. McDermott *et al.*, "A hybrid autoencoder and density estimation model for anomaly detection," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 717–726.
- [18] M. Sabokrou, M. Fathy, and M. Hoseini, "Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder," *Electronics Letters*, vol. 52, no. 13, pp. 1122–1124, 2016.
- [19] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.
- [20] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.
- [21] M. Nicolau, J. McDermott *et al.*, "Learning neural representations for network anomaly detection," *IEEE transactions on cybernetics*, vol. 49, no. 8, pp. 3074–3087, 2018.
- [22] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 622–637.
- [23] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.
- [24] P. Perera, R. Nallapati, and B. Xiang, "Ocgan: One-class novelty detection using gans with constrained latent representations," in *Proceedings*

- of the *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2898–2906.
- [25] D. M. Tax and R. P. Duin, “Support vector data description,” *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.
- [26] R. Chalapathy, A. K. Menon, and S. Chawla, “Anomaly detection using one-class neural networks,” *arXiv preprint arXiv:1802.06360*, 2018.
- [27] C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 251–261.
- [28] C. Kruegel, G. Vigna, and W. Robertson, “A multi-model approach to the detection of web-based attacks,” *Computer Networks*, vol. 48, no. 5, pp. 717–738, 2005.
- [29] W. Wang and X. Zhang, “High-speed web attack detection through extracting exemplars from http traffic,” in *Proceedings of the 2011 ACM symposium on applied computing*. ACM, 2011, pp. 1538–1543.
- [30] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Massegia, and X. Zhang, “Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks,” *Knowledge-Based Systems*, vol. 70, pp. 103–117, 2014.
- [31] K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest, “Learning dfa representations of http for protecting web applications,” *Computer Networks*, vol. 51, no. 5, pp. 1239–1255, 2007.
- [32] Y. Song, A. D. Keromytis, and S. Stolfo, “Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic,” 2009.
- [33] H. Lampesberger, P. Winter, M. Zeilinger, and E. Hermann, “An on-line learning statistical model to detect malicious web requests,” in *International Conference on Security and Privacy in Communication Systems*. Springer, 2011, pp. 19–38.
- [34] R. Kozik, M. Choraś, W. Holubowicz, and R. Renk, “Extreme learning machines for web layer anomaly detection,” in *International Conference on Image Processing and Communications*. Springer, 2016, pp. 226–233.
- [35] S. Zhang, B. Li, J. Li, M. Zhang, and Y. Chen, “A novel anomaly detection approach for mitigating web-based attacks against clouds,” in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. IEEE, 2015, pp. 289–294.
- [36] M. Zhang, S. Lu, and B. Xu, “An anomaly detection method based on multi-models to detect web attacks,” in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2017, pp. 404–409.
- [37] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin, “A deep learning method to detect web attacks using a specially designed cnn,” in *International Conference on Neural Information Processing*. Springer, 2017, pp. 828–836.
- [38] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, “An anomaly detection method to detect web attacks using stacked auto-encoder,” in *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE, 2018, pp. 131–134.
- [39] A. Gurina and V. Eliseev, “Anomaly-based method for detecting multiple classes of network attacks,” *Information*, vol. 10, no. 3, p. 84, 2019.
- [40] H. Mac, D. Truong, L. Nguyen, H. Nguyen, H. A. Tran, and D. Tran, “Detecting attacks on web applications using autoencoder,” in *Proceedings of the Ninth International Symposium on Information and Communication Technology*. ACM, 2018, pp. 416–421.
- [41] Y. Pan, F. Sun, Z. Teng, J. White, D. C. Schmidt, J. Staples, and L. Krause, “Detecting web attacks with end-to-end deep learning,” *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 1–22, 2019.
- [42] J. Liang, W. Zhao, and W. Ye, “Anomaly-based web attack detection: a deep learning approach,” in *Proceedings of the 2017 VI International Conference on Network, Communication and Computing*. ACM, 2017, pp. 80–85.
- [43] A. Moradi Vartouni, S. Mehralian, M. Teshnehlab, and S. Sedighian Kashi, “Auto-encoder lstm methods for anomaly-based web application firewall,” *International Journal of Information and Communication Technology Research*, vol. 11, no. 3, pp. 49–56, 2019.
- [44] T.-Y. Kim and S.-B. Cho, “Web traffic anomaly detection using c-lstm neural networks,” *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.
- [45] D. Jurafsky and J. H. Martin, “Speech and language processing. vol. 3, 37-43 pp,” 2018.
- [46] I. Kanaris, K. Kanaris, I. Houvardas, and E. Stamatatos, “Words versus character n-grams for anti-spam filtering,” *International Journal on Artificial Intelligence Tools*, vol. 16, no. 06, pp. 1047–1067, 2007.
- [47] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, 2015, pp. 649–657.
- [48] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [49] E. Even-Dar, Y. Mansour, and P. Bartlett, “Learning rates for q-learning,” *Journal of machine learning Research*, vol. 5, no. 1, 2003.
- [50] C. T. Giménez, A. P. Villegas, and G. Á. Marañón, “Http data set csic 2010,” *Information Security Institute of CSIC (Spanish Research National Council)*, 2010.
- [51] C. Raissi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire, “Web analyzing traffic challenge: description and results,” in *Proceedings of the ECML/PKDD*, 2007, pp. 47–52.
- [52] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.
- [53] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio, “Blocks and fuel: Frameworks for deep learning,” 06 2015.
- [54] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, and K. Franke, “Combining expert knowledge with automatic feature extraction for reliable web attack detection,” *Security and Communication Networks*, vol. 8, no. 16, pp. 2750–2767, 2015.
- [55] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.
- [56] A. M. Vartouni, M. Teshnehlab, and S. S. Kashi, “Saosa: Stable adaptive optimization for stacked auto-encoders,” *Neural Processing Letters*, vol. 52, no. 1, pp. 823–848, 2020.