Network Anomaly Detection Using LSTM Based Autoencoder

Mahmoud Said Elsayed
University College Dublin, Ireland
mahmoud.abdallah@ucdconnect.ie

Nhien-An Le-Khac
University College Dublin, Ireland
an.lekhac@ucd.ie

Soumyabrata Dev
University College Dublin, Ireland
soumyabrata.dev@ucd.ie

Anca Delia Jurcut
University College Dublin, Ireland
anca.jurcut@ucd.ie

ABSTRACT

Anomaly detection aims to discover patterns in data that do not conform to the expected normal behaviour. One of the significant issues for anomaly detection techniques is the availability of labeled data for training/validation of models. In this paper, we proposed a hyper approach based on Long Short Term Memory (LSTM) autoencoder and One-class Support Vector Machine (OC-SVM) to detect anomalies based attacks in an unbalanced dataset, by training the models using only examples of normal classes. The LSTM-autoencoder is trained to learn the normal traffic pattern and to learn the compressed representation of the input data (*i.e.* latent features) and then feed it to an OC-SVM approach. The hybrid model overcomes the shortcomings of the separate OC-SVM, in which its low capability to operate with massive and high-dimensional datasets. Additionally, we perform our experiments using the most recent dataset (InSDN) of Intrusion Detection Systems (IDSs) for SDN environments. The experimental results show that the proposed model provides higher detection rate and reduces the processing time significantly. Hence, our method provides great confidence in securing SDN networks from malicious traffic.

KEYWORDS

Security countermeasures, Anomaly detection detection, Deep Learning, LSTM, SDN, InSDN, Malicious traffic, Autoencoder

ACM Reference Format:
Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2020. Network Anomaly Detection Using LSTM Based Autoencoder. In *The 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWINet '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3416013.3426457

1 INTRODUCTION

Intrusion is the main cause of a security breach, where a malicious user can damage or steal vital information of the network system in a short time. Moreover, it can cause further financial losses and huge damages in IT critical infrastructure. For example, $350M and $70M are the sizes of the loss caused by Yahoo and respectively, Bitcoin data breach [18]. The intruder techniques have been evolved using sophisticated tools to create attacks exploiting vulnerabilities in the server protocols.

For these reasons, IDSs are essential tools to guarantee the availability, confidentiality, and integrity of the data. In general, IDSs are of two types: signature-based and anomaly-based detection systems. In signature-based techniques, malicious traffic can be detected based on the predefined rules. Although these techniques are widely used in commercial products due to their high detection rate and low false alarms, they cannot detect unknown or novel attacks. The attacker techniques are evolved every day, and adapted to make the anomalous activities similar to normal activities. Therefore, any change in the attack signature, even if it is very small, can help the attacker to bypass the defined rules easily. In addition, with the era of IoT and big data, an extensive number of rules are required to cover the daily attacks that can occur in the network system, making the database that stores the defined rules relativity large. Thus, the continuous updating of the database leads to a slowdown in the system performance. Anomaly-based IDSs gained the attention of the research community, due to their ability to discover novel attacks that are not used in the training models. Machine learning techniques are used in anomaly detection to build a model that can differentiate between anomalies events from the rest of the data. The anomaly detection aims to find the pattern in the data that deviates from other observation [4]. Hence, it is applied in several applications, including fraud detection [32], medical applications [1], video surveillance [20], data leakage prevention [5], and intrusion detection [17]. However, the notion of anomaly differs across the various applications and contexts. For example, an anomaly can be equipment failures in the industrial domain, credit card fraud in the fraud detection domain, suspicious movements in the video surveillance, etc. However, in the cybersecurity domain, the IT administrators narrow down the meaning of anomaly, and they consider any events that deviate from normal as anomalies, i.e., the anomaly is an indicator of malicious activities in the network traffic.

There are three different techniques to train the anomaly detection models, including supervised, unsupervised, and the semi-supervised manner. The concept of supervised learning is to train the detection model using labeled data for normal and for anomalies events. An example of supervised detection algorithms includes Support Vector Machine (SVM), Bayesian networks, artificial neural networks (ANN). Although, supervised learning techniques perform better compared to signature-based techniques, these methods fail to detect the zero-day attacks that can occur in the network daily. In addition, such techniques always need balanced and labeled data for training. However, the availability of labeled data is usually

Q2SWINet '20, November 16–20, 2020, Alicante, Spain

ACM ISBN 978-1-4503-8120-8/20/11.
https://doi.org/10.1145/3416013.3426457

a major issue and usually is not available for researchers. Besides, the labeling process can be a time-consuming task, error-prone, and tedious. Unsupervised methods do not require any labeled data for training. The goal of unsupervised methods is to organize the data into separate clusters, by grouping them based on similarities within each data cluster. One of the top limitations of using unsupervised learning is that the false alarms are relativity high since no answer labels are available. Besides, it always needs the expertise of the user to interpret and label the classes which follow that classification. Examples of unsupervised approaches include Hidden Markov Models, k-Means clustering, and Hierarchical clustering. In semi-supervised learning, only the normal data is used for training. Then, the trained model is applied in testing data, which includes both normal and anomalies events. In practice, it is easy to obtain a normal data class than to find anomalies data classes, where obtaining the anomalies data classes is costly in most application domains. This is the approach used by our proposed approach. In this paper, we consider a point anomaly detection to decide whether if the individual instance is anomaly compared to the remaining data.

1.1 Contribution

The main contributions of this paper are as follows – (a) We proposed a deep learning based on LSTM-autoencoder model for anomaly detection. The idea is to train the deep learning model using normal data only. In this case, the model is capable of replicating the input data at the output layer with a low reconstruction error. In the case of anomalies, the trained model fails to reconstruct anomalous instances, given a high error rate. The error is used as an indicator to differentiate between normal and anomalies instances. (b) We combined the OC-SVM algorithms with the LSTM-autoencoder to enhance the performance of the LSTM-autoencoder model. The lower dimension of the input data (i.e. extracted from the LSTM-autoencoder model) are trained with the OC-SVM algorithm to achieve better classification results, whilst significantly reducing the training time. (c) We used the recently generated dataset InSDN [9] to ensure an accurate evaluation of the proposed approach, since InSDN dataset is representative of the attacks specific to SDN environments. The dataset used for evaluation is critical since the performance of IDSs relies on the quality of the training datasets. This a significant contribution, since the current IDSs are based on fundamentals that are not representative of SDN environments or suffer from several shortcomings related to the intrusion dataset generated for the classification process.(d) The computational overhead of the proposed DL model is also evaluated to verify the model performance for real-time intrusion detection. Based on the experiment results, the proposed model is able to identify anomalies with a high detection rate and solve the problem of unbalanced and unlabeled datasets.

The structure of the remaining part of the paper is as follows: Section 2 provides a systematic review of the various learning models for detecting attacks in network traffic. We propose our deep learning model for encoding the input feature space and the binary classification framework in Section 3. Section 4 presents our benchmarking results in the publicly available InSDN dataset. Finally, Section 5 concludes the paper and discusses our future work.

2 ABNORMAL EVENT DETECTION

In this section, we discuss the various machine learning and deep learning models that have been proposed for attack detection in network traffic. With the advent of deep learning based models, the accuracy of attack detection has further improved. The deep learning based methods are useful, because the discriminative features are automatically generated, without the use of generating hand-crafted features.

2.1 Related Work

Most of the current works based on anomaly detection methods applied ANN for classification tasks. The labeled data is used during the training stage, and then the learned model is applied on testing data to classify it into one of the classes. In [28], the authors presented a flow-based detection approach using Self Organizing Maps (SOMs) classifier in the network environment. The evaluated accuracy of the proposed model in the testing phase is 74.67%. For overall evaluation, the precision, recall, and f-measure obtained from the conducted model are 83%, 76%, and 75%, respectively.

Latah *et al.* [19] proposed a five-stage hybrid classifier system to enhance the detection rate against malicious traffics inside the network. The model combines three different machine learning classifiers, including the K-Nearest Neighbor approach (KNN), Extreme Learning Machine (ELM), and Hierarchical Extreme Learning Machine (H-ELM). The overall accuracy of the presented approach is 84.29%, while the percentage of precision, recall, and F1-score is 94.18, 77.18, and 84.83, respectively.

Prasath *et al.* [22] proposed a Novel Agent Program (NAP) framework to secure a communication model of the virtual switches in the network. The meta-heuristic Bayesian network classification (MHBNC) approach is used to classify the incoming packets into normal or attack traffic. The proposed MHBNC model has achieved an overall accuracy of 82.99%. Besides, the realized precision, recall, and f-score is 77%, 74%, and 75%, respectively.

One of the main advantages of the aforementioned techniques is the ability to handle high-dimensional data sets with high performance. However, these approaches mainly rely on labeled and balanced data, and this is an issue. The majority of real data are unbalanced, where the anomalies data are often challenging and less frequently to obtain compared to normal data.

On the other side, the autoencoder, which considers a specific kind of feed-forward neural networks, gained the research community's attention. The autoencoder is mainly applied in outlier based anomaly detection rather than classification problems. One of the first studies that involved autoencoder for outlier detection was proposed by Hawskin *et al.* [12] and it is used widely in the research community. In recent years, the number of studies using autoencoder as a complementary algorithm for feature reduction tasks has increased. The autoencoder achieves great success in generating abstract features of high dimensional data. It can significantly increase the anomaly detection accuracy in comparison to linear and kernel PCA [24]. It can detect subtle anomalies that the linear PCA fails to detect. Furthermore, the autoencoder is easy to train and does not require complex computation like kernel PCA. A comprehensive study of using the autoencoder in anomaly detection approaches is discussed in [3].

In our previous work [8], we used the reconstruction error as a threshold to detect anomalies on the NSL-KDD dataset. Although the obtained results are significantly high, it seems that the experiment results are more specific for the NSL-KDD dataset. However, the traces of the NSL-KDD dataset were generated two decades ago and the distribution of normal and malicious traffic significantly deviates from each other. So, the simple threshold is an excellent choice to separate between different boundaries. In contrast, in modern traffic, the attacker can use very sophisticated tools to generate new attack classes that are extensively similar to legitimate traffic. Therefore, some anomaly error rates are quite close to the standard legitimate error rate, and it is not easy for anomaly detection systems to detect them with a high performance rate [27]. Thus, the simple threshold is insufficient, especially for high dimension data, as the reconstruction errors are not linearly separable. In addition, the high similarity in some normal and abnormal traffic creates data samples with a small distance between them, which sophisticates the use of the simple threshold in attack detection systems.

2.2 Limitations in the Traditional Machine Techniques

The machine learning and shallow learning techniques, such as SVM, NB and RF have been widely deployed in intrusion detection systems to recognize attack threats [10, 16, 26]. These approaches attempt to learn the feature representation in network traffic data for an effective classification. However, it is not easy to manually handcraft and extract the discriminatory features in intrusion detection systems. Firstly, the nature of the attacks evolve everyday, and the attacker's techniques change with time. Secondly, the features which are extracted for one category of attack, may not necessarily be suitable for other attack classes. As a result, selecting the significant features to identify the attack from network traffic is a cumbersome task. Therefore, existing attack detection techniques fail to discover all types of attacks. Furthermore, there is a high degree of non-linearity in the dataset, and therefore the traditional machine learning based methods fail to classify the normal and malicious data types [7]. Elsayed *et al.* further established this fact in [7] by generating the Andrews curve for the NSL-KDD dataset. The Andrews curve represents a high-dimensional feature space in the form of a finite Fourier series. This provides a visual understanding of the internal structure of the dataset. Figure 1 shows the Andrews curve for the NSL-KDD dataset. Each curve in Fig. 1 represents an observation in the dataset. We observe that the two labels are not clearly grouped in two separate streams. The legitimate and malicious data curves are tangled with each other, indicating a high-degree of inherent non-linearity in the feature space. Therefore, traditional machine learning techniques fail to capture the non-linearity in such datasets.

Hence, in this work we propose a deep learning technique using LSTM-autoencoder to model the normal traffic data. This assists in proposing a robust framework for detecting attacks in SDN network traffic. Unlike other machine learning approaches, our proposed technique can automatically learn the discriminatory features from the network traffic data. However, we formulate our problem as a binary classification problem, wherein we classify any network data into *normal* and *malicious* type.

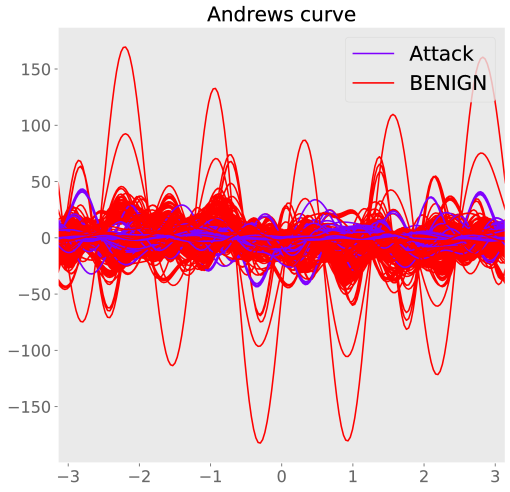

Figure 1: We demonstrate the Andrews curve for the InSDN dataset. We plot the legitimate and malicious observations in green and red curves respectively.

3 PROPOSED MODEL

This section presents the framework elements and the system architecture of our proposed IDS model.

3.1 Autoencoder

An autoencoder is an artificial neural network that applies back-propagation, to produce the output vector similar to the inputs. It compresses the input data into a lower-dimensional space, then reconstructs the original data again from this representation. It uses a non-linear activation function and multiple layers to learn the non-linear relation in the data. A simple illustration of the autoencoder model architecture is shown in Fig. 2.

The Autoencoder is considered an unsupervised learning technique since it does not require a separate label value to train. In practice, the autoencoder is composed of two phases: the encoder and the decoder parts. The main objective of the encoder phase is to reduce the dimensions of the input data $\mathbf{X}$ according to the equation 1.

$$Z = \sigma(WX + b) \tag{1}$$

Here, $\mathbf{Z}$ is the latent dimension, σ is the activation functions, $\mathbf{W}$ is the weight, and $\mathbf{b}$ is the bias vector. In the same manner, the decoding phase is trained according to the equation 2 in order to obtain the output data similar to the original space, but with different bias, weight and possibly activation functions.

$$X' = \sigma'(W'Z + b') \tag{2}$$

The main goal of the autoencoder is to make the output vector similar to the original space, by minimizing the reconstruction error between them. The reconstruction error can be obtained using a cross-entropy function or sum of squared errors (SSE). In this paper, we used SSE to calculate the reconstruction error according to the equation 3

$$SSE = \sum_{i=1}^{n} (\mathbf{X}'_i - \mathbf{X}_i)^2 \tag{3}$$

The decoder part regenerates the initial data based on the encoder output. To achieve the dimensional reduction and generate the compressed feature vector of input data, the code layer [13] is used at the center of the autoencoder structure. The code layer can be utilized for classification activities or combined with another stacked autoencoder [30].

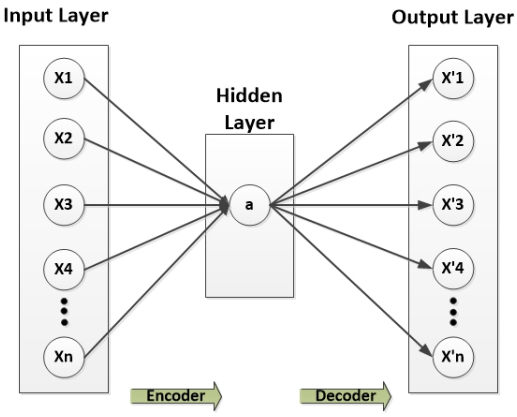

Figure 2: An example of a single autoencoder.

3.2 RNN to LSTM

Recurrent Neural Network (RNN) is a class of artificial neural network with backward connections, where the output from a network layer is returned to either that layer or to a previous network layer.

RNN can address the problem of traditional feed-forward neural networks [31]. As a result, it can create much powerful models with high classification accuracy. RNN is widely applied in different domain applications such as language processing and speech recognition. Unlike the feed-forward neural networks, the cyclic connections of the RNN can be effectively used for modeling sequences [14]. In RNN, for the given input vector sequence $X = (x_1, x_2, x_3, ..., x_t)$, we can compute the hidden vector $Z = (z_1, z_2, z_3, ..., z_t)$ and output vector sequence $F = (f_1, f_2, f_3, ..., f_t)$ at time t using Eq.4 and Eq.5, respectively.

$$z_t = \sigma(W_{xz}X_t + W_{zz}z_{t-1} + b_h), \tag{4}$$

$$f_t = W_{zf}h_t + b_f, \tag{5}$$

Here, σ is the activation function, W is the weight, b is the bias and z_{t-1} is the state at time $t-1$.

In this work, we use the RNN based on the nature of the input data, where the temporal correlations of network traffic often generate time-series data [29]. For this reason, we used the RNN-based approach to solve the problem of simple feed-forward neural networks, since RNN considers the previous output and the current input at each stage. In addition, RNN has been applied efficiently in the anomaly detection for traditional networks [2][21]. Training the model with such methods can minimize the loss and further, it can provide high performance.

The main issue in RNN is the vanishing gradient problem. The gradient is used to update the weight values of the learned model. However, in case the gradient is very small, the model can not learned efficiently. Thus, layers that get a small gradient update in RNN will stop learning, and usually, this issue happens in earlier layers. The LSTM algorithm [11] was explicitly proposed as a solution to avoid the vanishing gradient problem. The LSTM uses the mechanism of gates to regulate the flow of information. LSTM composites of three control gates: forget, output, and input gates. The forget gate keeps a fraction of previous state information, while output gate is responsible for choosing how much of an information we output and the input gate is responsible for getting new information.

3.3 One-Class SVM

OC-SVM [25] approach is a special case of support vector and widely used to discover anomalies in an unsupervised fashion. It is trained only on the 'normal' data to learn the boundaries of these points. Then, it is able to classify any points that lie outside the boundary i.e. outliers. The main difference between the standard SVM and one-class SVM is that the OC-SVM provides a hyperparameter "nu", which is used to control the sensitivity of the support vectors, instead of the normal hyperparameters like C in the standard SVM, which is used for tuning the margin.

Here, let the training samples $(x_1, x_2, x_3, ..., x_l)$, belonging to one known class X (i.e. "normal driving"). Let ϕ is a kernel map function that transform the training samples into another space. We need to solve the following objective function of one-class SVM to separate the data set from the origin [23]:

$$min \frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i=1}^{l}\varepsilon_l - \rho, \tag{6}$$

$$subject\ to: w\phi(x_i) \geq \rho - \varepsilon_i,\ i = 1, 2, 3, 4, ..., l,\ \varepsilon_i \geq 0 \tag{7}$$

where w is a decision hyperplane, ρ is the bias term and ε_i is a Nonzero slack variables. The meta-parameter $\nu \in (0, 1)$ is used to control the number of samples contained in the hyper sphere. The decision function corresponding to w and ρ is:

$$f(x) = w\phi(x) - \rho \tag{8}$$

The main objective is to find a hyper sphere, which contains most of the training samples obtained consequently from the target region. After training, the decision boundary may allow us to choose the most appropriate candidate region.

3.4 InSDN Dataset

The performance of the IDSs techniques relies on the quality of the training datasets. One of the main challenges in the deployment of the detection mechanisms is the lack of available up-to date real-world datasets. The main reason for the lack of public datasets for the intrusion detection domain returns to privacy and legal issues. In this work, we are using the InSDN dataset to evaluate our proposed deep learning model. The InSDN dataset was generated to overcome the shortcomings of the existing datasets in the context of SDN network [9].

The InSDN dataset contains various attack scenarios and attack classes such as DoS, DDoS, Web attacks, Password-Guessing, Botnet, Exploitation, and Probe attacks. Besides, the normal traffic in InSDN includes various popular application services such as HTTPS, HTTP, DNS, Email, FTP, SSH. The source of attacks in the dataset comes from internal and external network to mimic the real attack scenarios. It contains more than 80 statistical features in `CSV` file format such as Protocol, Duration, Number of bytes, Number of packets, etc. The total number of dataset instances are 343,939 for normal and attack traffic, where the normal data brings a total of 68,424, and attack traffic contains 275,515 instances.

Using InSDN dataset for the evaluation of our proposed model it provides more accurate results, since the nature of attacks in the SDN is different from those commonly affecting the conventional networks. When the OpenFlow switch receives any unknown flow packets, it will send these flows to the SDN controller in the form of `packet-In` message for further processing. Since the normal and malicious traffic is forwarded to the SDN controller for decision making, the attack traffic mimics the same normal behavior. Moreover, the centralized view of the SDN network and separation of the data plane from the control plane creates a new opportunity for the attacker to carry out various types of attacks compared to the conventional network. These attacks are not easy to detect as the intruder is connected to the victim server in an authorized manner. As a result, using such dataset for the model evaluation can be a good indicator to reflect the real-world scenario. In addition, the InSDN dataset does not contain any redundant records, which prevents the learner model to bias towards the most frequent records.

3.5 Dataset Preparation

In this paper, we focus our attention on a binary classification problem, and do not delve further into classifying the various types of attacks. The observations belonging to any attack class are categorized as anomalous traffic data. The first phase before training the IDS model is preparing the dataset for proper use. Few steps are taken for pre-processing the entering flows, as follows:

- The generated dataset contains the socket information such as Source IP, Destination IP, flow ID, etc. We remove all socket features to avoid the overfitting problem, where such data can be changed from network to network. The final dataset includes 77 various features, besides the traffic category. Table 1 shows the flow samples that are used for training and testing during our model training. We randomly selected some samples from the entire dataset, where the size of training and testing records are not very large. Thus, the time is taken for the model training can stay reasonable.
- The features have different ranges so they need to be standardized.
- We use one-hot encoding to convert the labeled string to numerical values. In this model, we consider only binary classification to identify the malicious and normal traffic from input data. Therefore, we are encoding the normal string to a binary value of 0 and respectively, all malicious traffic of 1.

3.6 Modeling the normal traffic data

This section introduces the proposed architecture model for detecting network attacks. We know that deep learning can assist us in representing large-scale network traffic with a more discriminatory feature space. Such technique uses multiple processing layers to model the input feature properly. This is advantageous as compared to traditional hand-crafted feature descriptors, because deep learning techniques can automatically extract the discriminatory features, as compared to manually generating the features. Our proposed approach can estimate a good representation of the input feature space. Figure 3 describes our proposed architecture to model the normal network data.

We use the RNN and autoencoder architectures based on the nature of the input data, where the temporal correlations of network traffic often generate time-series data [29]. For this reason, we used the RNN-based approach to solve the problem of simple feed-forward neural networks since RNN considers the previous output and the current input at each stage. In addition, RNN has been applied efficiently in the anomaly detection for traditional networks [2, 6, 21]. Training the model with such methods can minimize the loss and further, it can provide high performance. Additionally, the autoencoder has the advantage in number of classification problem. The reason that we decided to use the autoencoder in our proposed model for anomaly detection is the fact that the autoencoder is trying to learn the best parameters to reconstruct the input at the output layer. Moreover, we adapted the LSTM algorithm for our model to solve the issues of the standard RNN technique, such as vanishing and exploding gradient problems [15].

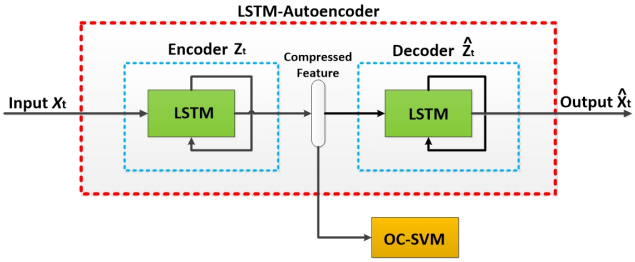

Figure 3: Proposed model to encode the input features. We use blocks of encoder and decoder comprising LSTM layers.

Additionally, our model uses LSTM with autoencoder to learn the representations of the network dataset in a simi-supervised fashion, as depicted in Fig. 4. It contains multiple layers of encoder

#	Type	No. of Training Instances	No. of Testing Instances
1	Normal	57956	10468
2	Probe	12586	2639
3	DoS-Network\Transport layer	3160	612
4	DoS-Application layer	18462	13166
5	DDoS	9440	503
6	Brute_force_attack	1117	288
7	Botenet	164	-
8	Web Application Attack	174	18
9	Exploitation (U2R)	14	3
Total	-	**103,073**	**27,697**

Table 1: Traffic Flows for Training and Testing.

and decoder stages and each stage consists of multiple LSTM units. The input data $\mathbf{X}_t$ is encoded via the encoder block to generate a fixed range feature vector $\mathbf{Z}_t$. The input data $\mathbf{X}_t \in \mathbb{R}^{77\times1}$ is the initial encoded feature vector generated from the dataset. We set `timestamp = 1` for our LSTM blocks. We used the LSTM blocks for individual events, and not for time series. The encoder block sequentially reduces the dimension of the 77 dimension initial feature vector. The dimensions are reduced to 128, 64, 32, and 16, after the first, second, third and fourth layers of the encoder respectively. The final encoded feature vector $\mathbf{Z}_t \in \mathbb{R}^{16\times1}$ represents the compressed input data. The low-dimensional representation of the input data $\mathbf{Z}_t$ is trained with OC-SVM for anomaly classification. The model is trained using normal traffic only, so the malicious traffic will be considered as outliers.

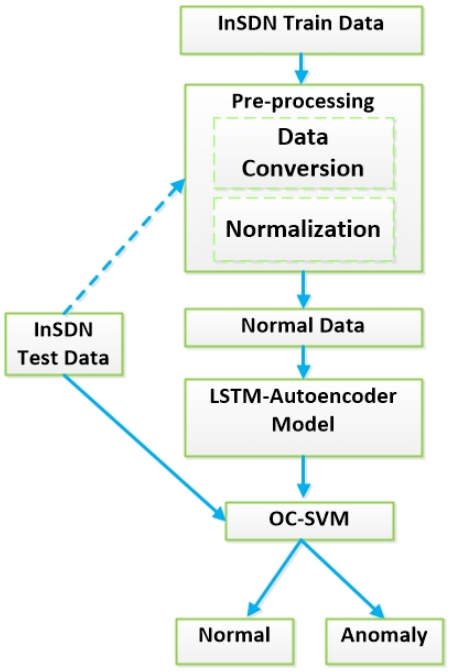

Figure 4: The diagram flow of the proposed method using InSDN data set.

The encoded data is then fed into the decoder block for generating the output feature vector. We represent the input feature vector of the decoder block as $\widehat{\mathbf{Z}_t}$. The layers in the decoder block are arranged in the reverse order as that of the encoder layers. The encoded features $\widehat{\mathbf{Z}_t}$ are then fed via a series of LSTM blocks to generate the output feature vector $\widehat{\mathbf{X}_t}$. The dimensions are increased to 16, 32, 64, 128, after the first, second, third and fourth layers of the decoder respectively. Finally, the final layer of the decoder block is fed to a fully connected layer to generate the output feature vector $\widehat{\mathbf{X}_t}$. We attempt to reconstruct this output feature vector $\widehat{\mathbf{X}_t}$ to be as close as the input feature vector $\mathbf{X}_t$. We use the mean square error (MSE) to calculate the estimation error between input data $\mathbf{X}_t$ and output representation $\widehat{\mathbf{X}_t}$.

The LSTM-autoencoder network is used to model the normal traffic data using the discriminatory feature $\widehat{\mathbf{X}_t}$. The reconstruction error for normal traffic data will be less as compared to that of anomalous traffic data. This behavior will greatly help in detecting the anomalous traffic since its corresponding error value will be considerably higher.

4 EVALUATION AND RESULTS

This section details the evaluation process of our technique and shows that our method provides great confidence in securing the networks from malicious traffic.

4.1 Loss trend of our proposed model

We use the generated feature $\mathbf{X}_t$ to train our model, such that the reconstruction loss is minimum. We use a learning rate of 0.0001, batch size of 32, tanh activation function, and train the model using Adam optimizer. We train our model for 100 epochs. Table 2 summaries the choice of the different hyper-parameters. Figure 5 describes the trend of training and validation loss over the number of epochs. We observe that the loss trend is similar for training and validation sets, and converges after a few tens of epochs.

4.2 Performance Metrics

We use precision, recall, f-Score, and accuracy to evaluate our model performance. The mathematical representation of these metrics are calculated as follows:

Parameters	Best Values
Hidden layers	4
Hidden layer size (neurons)	128, 64, 32, and 16
Optimizer	Adam
Loss function	MSE
Activation function	Tanh
Learning rate	0.0001
Number of epochs	100
Batch size	32

Table 2: We mention the values of the several hyper-parameters. We conducted different experiments to get the best values of hyper-parameters for model initiation. During these experiments, we change the value of learning rate, hidden layers size, epochs, and batch size that provide a high accuracy rate.

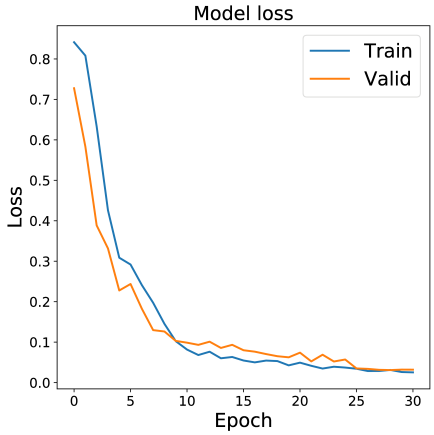

Figure 5: Trend of training and validation loss over the number of epochs.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

where TP (True Positive) represents the number of instances correctly classified as an attack; TN (True Negative) represents the number of instances correctly classified as normal; FP (False Positive) represents the number of instances incorrectly classified as an attack; FN (False Negative) represents the number of instances incorrectly classified as normal.

4.3 Anomaly detection with a simple Threshold

We train our deep learning model using traffic data that are labeled as normal. We compute the ℓ_2-norm error between the original feature $\mathbf{X}_t$ and the output feature $\widehat{\mathbf{X}_t}$ in order to compute the reconstruction error. The ℓ_2-norm error $e = \|\mathbf{X}_t - \widehat{\mathbf{X}_t}\|^2$ will be low for *normal* traffic data, and high for *anomalous* traffic data. Therefore, we use a fixed threshold in the reconstruction error for the binary classification of *normal* and *anomalous* traffic data. The threshold value is used as a decision boundary for detecting anomalous data. The observations that have a reconstruction error greater than the threshold will be classified as anomalous, whereas the ones with reconstruction error less than the threshold as normal traffic data.

We illustrate the efficacy of the proposed approach by reporting different threshold values and represent their impact on precision, recall, F-score, and accuracy. Table 3 summarizes the performance of different threshold values in terms of evaluation metrics. The best performance is obtained at a threshold value 0.07. Then the evaluation metrics drop dramatically with the increase in the threshold value.

Threshold	Precision	Recall	F1-measure	Accuracy
0.05	0.652	0.986	0.785	0.664
0.06	0.6919	0.986	0.813	0.718
0.07	**0.7111**	**0.983**	**0.825**	**0.741**
0.08	0.690	0.847	0.7611	0.669
0.09	0.667	0.760	0.710	0.615
0.1	0.587	0.539	0.562	0.477
0.2	0.584	0.438	0.500	0.456
0.3	0.202	0.065	0.099	0.257

Table 3: Evaluation Metrics with different threshold values.

However, using the reconstruction error as an anomaly threshold cannot significantly separate the normal and malicious data. The high degree of similarity in some malicious and legitimate traffic makes the reconstruction error rates for both traffic are relatively close to each other i.e. are not linearly separated. To overcome this gap, we integrate the One-Class SVM algorithm with the LSTM-Autoencoder to better characterize the network traffic; hence, the detection rate can significantly improve.

4.4 LSTM-Autoencoder-OC-SVM for Attack Detection

As mentioned in the previous section, the simple threshold does not perform well in our experiments. The complexity of attacks in the employed dataset and the high similarity of some attack traffic with legitimate one are the main concern to use the simple threshold for anomaly detection works, especially in SDN environments.

This section analyzes the detection performance of the LSTM-AE-OC-SVM approach. In this paper, we focus our attention on a binary classification problem, wherein we classify each observation as *normal* and *anomalous* traffic data. We analyse the precision, recall, F-score, and accuracy values for all methods considered.

The results are presented in Table 4. For the OC-SVM model, the parameters gammas=0.001, nu=0.4 and a Radial Basic Function (RBF) kernel are choosen for the experiment. Our approach has the best performance in terms of F-score and accuracy values. The OC-SVM technique fails to have competitive F-score and accuracy values.

Algorithm	Precision	Recall	F1-measure	Accuracy (%)
OC-SVM	0.89	0.93	0.91	87.5
LSTM-Autoencoder-OC-SVM	0.93	0.93	0.93	90.5

Table 4: The Evaluation Metric Comparison. We report the precision, recall, F-score and accuracy for the different both algorithms.

We also check the efficacy of our proposed method in terms of computational time. The computational time is very important to evaluate a classifier's performance, especially with the era of big data since massive amount of data is needed for the classification in real-time. The table 5 represents the training and testing times of the OC-SVM algorithm and hybrid approach. We observe the consumed time by the OC-SVM algorithm is significantly high compared to the hybrid approach for both training and testing.

Algorithm	Training Time (s)	Testing Time (s)
OC-SVM	479.748	38.355
LSTM-Autoencoder-OC-SVM	147.548	13.546

Table 5: The Training and Testing Time for Both Algorithms

4.5 Receiver Operating Characteristic (ROC)

Further, we use the receiver operating characteristic (ROC) curve for checking the performance of the proposed approach. The ROC curve represents the relation between false positive and true positive rates. The area under the curve indicates the efficacy of the binary classifier. The binary classifier gives perfect measures when the area under curve (AUC) is near to the 1. In contrast, the model has the worst measures during AUC near to the 0. The AUC value of the presented model are shown in Figure 6. We obtained the value of 0.906, which indicates that our model can successfully separate 90.6% of positive and negative rates.

5 CONCLUSION AND FUTURE WORK

Network data can often be compromised because of malicious attacks initiated by intruders. A good practice to protect against these attacks is to deploy machine learning based frameworks to detect anomalies caused during the attacks. In this paper, we highlighted the existing problems in exiting techniques and proposed solutions to address them. We proposed a deep learning framework based on LSTM-autoencoder and OC-SVM that can model the normal traffic data efficiently. Our experiments shows that our proposed model can efficiently detect the anomalies presented in network traffic data. In our future work, we plan to apply the proposed IDS framework in one or more realistic network settings to evaluate its performance in real-world scenarios and test its impact with regard to latency. We also plan to extend the binary classification problem into a multi-class classification problem, in order to identify the type of network attacks.

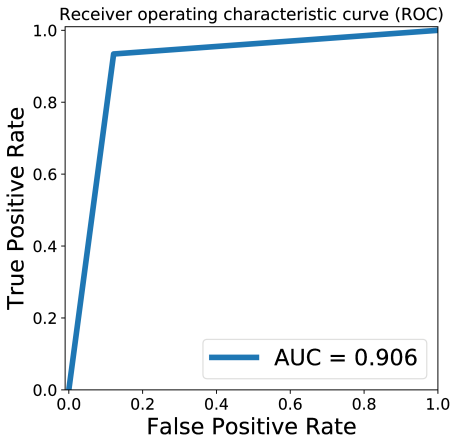

Figure 6: Receiver Operating Curve (ROC) of our proposed approach.

6 ACKNOWLEDGMENTS

This research is funded by the School of Computer Science, University College Dublin, Ireland. Dr. Anca Jurcut is involved in the work conducted with COST Action 17124 DigForAsp, supported by COST (European Cooperation in Science and Technology) www.cost.eu".

REFERENCES

[1] Mohammed Abbass, Ki-Chul Kwon, Nam Kim, Safey A Abdelwahab, Nehad Haggag, Fatma Ibrahim, Yasser Mahrous, Ahmad Seddik, Ali Khalil, Zeinab Elsherbeeny, et al. 2020. Anomaly Detection from Medical Signals and Images Using Advanced Convolutional Neural Network. (2020).

[2] L. Bontemps, V.L. Cao, J. McDermott, and Nhien-An Le-Khac. 2016. Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks. *In: Dang T., Wagner R., Küng J., Thoai N., Takizawa M., Neuhold E. (eds) Future Data and Security Engineering. FDSE 2016. Lecture Notes in Computer Science, vol 10018. Springer, Cham* (2016).

[3] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[5] Elisa Costante, Davide Fauri, Sandro Etalle, Jerry Den Hartog, and Nicola Zannone. 2016. A hybrid framework for data loss prevention and detection. In *2016 IEEE Security and Privacy Workshops (SPW)*. IEEE, 324–333.

[6] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. [n. d.]. Ddosnet: A deep-learning model for detecting network attacks. In *21ST IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS (IEEE WOWMOM 2020), Ireland*. IEEE.

[7] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2019. Machine-Learning Techniques for Detecting Attacks in SDN. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 277–281.

[8] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2020. Detecting Abnormal Traffic in Large-Scale Networks. In *2020 IEEE International Symposium on Networks, Computers and Communications (ISNCC'20)*. IEEE.

[9] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. 2020. InSDN: A Novel SDN Intrusion Dataset. *IEEE Access* 8 (2020), 165263–165284.

[10] S. Garg and S. Batra. 2017. A novel ensembled technique for anomaly detection. *International Journal of Communication Systems* 30, 11 (2017), e3248.

[11] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).

[12] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. 2002. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 170–180.

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[14] F. Jiang, Y. Fu, B. B. Gupta, F. Lou, S. Rho, F. Meng, and Z. Tian. 2018. Deep learning based multi-channel intelligent attack detection for data security. *IEEE transactions on Sustainable Computing* (2018).

[15] J. Kim, J. Kim, H. L. T. Thu, and H. Kim. 2016. Long short term memory recurrent neural network classifier for intrusion detection. In *Proc. International Conference on Platform Technology and Service (PlatCon)*. IEEE, 1–5.

[16] F. Kuang, W. Xu, and S. Zhang. 2014. A novel hybrid KPCA and SVM with GA model for intrusion detection. *Applied Soft Computing* 18 (2014), 178–184.

[17] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. 2019. A survey of deep learning-based network anomaly detection. *Cluster Computing* (2019), 1–13.

[18] D. Larson. 2016. Distributed denial of service attacks–holding back the flood. *Network Security* 2016, 3 (2016), 5–7.

[19] M. Latah and L. Toker. 2018. An Efficient Flow-based Multi-level Hybrid Intrusion Detection System for Software-Defined Networks. *arXiv preprint arXiv:1806.03875* (2018).

[20] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu. 2019. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Transactions on Industrial Informatics* 16, 1 (2019), 393–402.

[21] N. Nguyen Thi, V.L. Cao, and Nhien-An Le-Khac. 2017. One-Class Collective Anomaly Detection Based on LSTM-RNNs. *In: Hameurlain A., Küng J., Wagner R., Dang T., Thoai N. (eds) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVI. Lecture Notes in Computer Science, vol 10720. Springer, Berlin, Heidelberg* (2017).

[22] M. K. Prasath and B. Perumal. 2019. A meta-heuristic Bayesian network classification for intrusion detection. *International Journal of Network Management* 29, 3 (2019), e2047.

[23] Saina Ramyar, Abdollah Homaifar, Ali Karimoddini, and Edward Tunstel. 2016. Identification of anomalies in lane change behavior using one-class SVM. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 004405–004410.

[24] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. 4–11.

[25] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. 2000. Support vector method for novelty detection. In *Advances in neural information processing systems*. 582–588.

[26] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad. 2019. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications* 12, 2 (2019), 493–501.

[27] T Tang, L Mhamdi, S Zaidi, F El-moussa, D McLernon, and M Ghogho. 2019. A Deep Learning Approach Combining Auto-encoder with One-class SVM for DDoS Attack Detection in SDNs. In *Proceedings of the International Conference on Communications and Networking*. IEEE.

[28] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho. 2016. Deep learning approach for network intrusion detection in software defined networking. In *Proc. International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 258–263.

[29] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho. 2018. Deep recurrent neural network for intrusion detection in SDN-based networks. In *Proc. 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 202–206.

[30] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, Dec (2010), 3371–3408.

[31] C. Yin, Y. Zhu, J. Fei, and X. He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961.

[32] Junyi Zou, Jinliang Zhang, and Ping Jiang. 2019. Credit Card Fraud Detection Using Autoencoder Neural Network. *arXiv preprint arXiv:1908.11553* (2019).