Journal of Big Data

RESEARCH Open Access

Intrusion detection systems using long short-term memory (LSTM)

FatimaEzzahra Laghrissi[1*], Samira Douzi[2*], Khadija Douzi[1*] and Badr Hssina[1*]

*Correspondence: fatimaezzahra.laghrissi@etu.fstm.ac.ma; s.douzi@um5r.ac.ma; khadija.douzi@fstm.ac.ma; badr.hssina@fstm.ac.ma
[1] FSTM, University Hassan II, Casablanca, Morocco
[2] FMPR, University Mohammed V, Rabat, Morocco

Abstract
An intrusion detection system (IDS) is a device or software application that monitors a network for malicious activity or policy violations. It scans a network or a system for a harmful activity or security breaching. IDS protects networks (Network-based intrusion detection system NIDS) or hosts (Host-based intrusion detection system HIDS), and work by either looking for signatures of known attacks or deviations from normal activity. Deep learning algorithms proved their effectiveness in intrusion detection compared to other machine learning methods. In this paper, we implemented deep learning solutions for detecting attacks based on Long Short-Term Memory (LSTM). PCA (principal component analysis) and Mutual information (MI) are used as dimensionality reduction and feature selection techniques. Our approach was tested on a benchmark data set, KDD99, and the experimental outcomes show that models based on PCA achieve the best accuracy for training and testing, in both binary and multiclass classification.

Keywords: Intrusion detection systems, Deep learning, LSTM, PCA, Mutual information

Introduction

With more than 26 billion connected devices in 2019, network attacks are changing constantly, and cyber threats become serious issues due to the over-reliance of government, military and commercial bodies on the internet for their day to day activities. IDSs are one of the critical and vital components of the security infrastructures intended to improve security in the computer systems. Over the last years, many machine learning models have been developed and evaluated. Some IDS used classification algorithms like Decision tree, SVM, K-nearest, while others use feature selection. Almost all IDS based on machine learning algorithms uses shallow learning (single feed forwards networks.), which rely on manual feature engineering to craft features of the ML model [1]. Moreover, Shallow learning proved his inability of solving real-time environmental problems, because of the huge amount of data entries. For that reason, models based on deep learning, such as a recurrent neural network (RNN), variational autoencoder (VAE), and long short-term memory (LSTM), have increased in the past few years.

Furthermore, the number of features extracted from raw network data, which an IDS needs to examine, is usually large even for a small network. Moreover, most of the extracted data are unimportant and noisy, which leads to the presence of irrelevant features that will deteriorate the performance of the classifier. Hence, dimensionality reduction algorithms, such as PCA (Principal component analysis) and Mutual information are crucial, to select informative data.

In this research, our contribution consists of implementing three models, namely: LSTM, LSTM-PCA, and LSTM-MI (Mutual Information). As shown in Fig. 1, after preprocessing the well-known dataset KDD99, we applied three different approaches, based on Long short-term memory classifier. LSTM without any dimensionality reduction (using all KDD99 features), applying Principal Component Analysis and employing Mutual Information. These approaches were tested on binary and multiclass classification, and the result showed that models based on PCA obtained the best results.

Related works

From the first intrusion detection system introduced by Denning [2], studies applied multiple intrusion detection algorithms. Since traditional classification algorithm need to manually extract features, deep learning techniques proved their effectiveness in this type of problem. Therefore, many researchers have focused their efforts on deep learning techniques to create powerful IDSs.

Anand [3] used an improved genetic K-means++ algorithm and IGKM, on a subset of the KDD99 dataset to create an intrusion detection system. IGKM gave a better accuracy than K-means++. Ma [4] used a Deep Neural Network (DNN) to classify attack type and Spectral Clustering as a feature extractor. The DNN network had a better accuracy than SVM (support-vector machine), BPNN (back propagation neural network) and RF (Random Forest). Nikolov [5], proposed a recurrent neural network classifier namely long-short term memory (LSTM), the authors tested the solution on the NGID-DS dataset for binary classification (Only HTTP Attack). Jayaprakash [6], introduced an anomaly based detection mechanism that implements Role based Access control (RBAC). A new data structure called Octraplet is used for storing the SQL queries. This system uses the Naive Bayes Classifier which is a supervised Machine Learning method for Detecting anomalous queries. Kang [7] proposed an effective IDS on for the in-vehicle

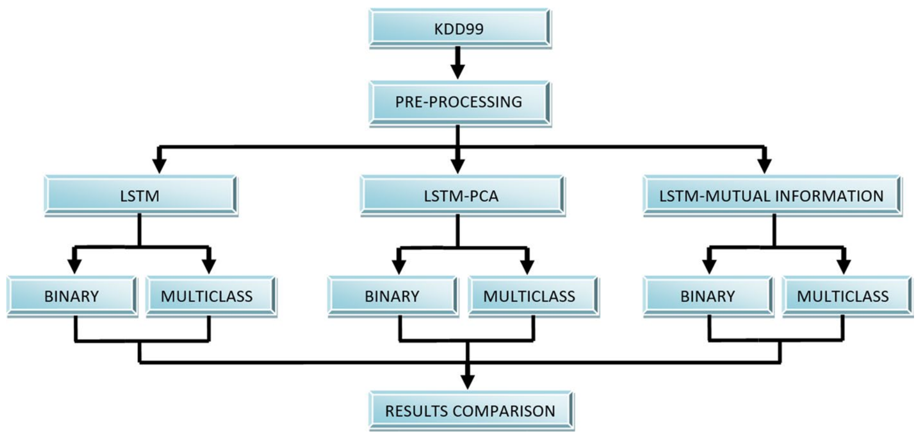


Fig. 1 Models compared in this research

network. The system is used to detect attacks in a controller area network (CAN) bus. The model loads the over pre-training belief (DBN) an enhancement in the detection accuracy. Chkirbene [8], proposed two models for intrusion detection and classification, Trust-based Intrusion Detection and Classification System (TIDCS) which reduces the number of features in the input data based on a new algorithm for feature selection and TIDCS-A which is a dynamic algorithm to compute the exact time for nodes cleansing states and restricts the exposure window of the nodes, this method is tested on NSLKDD and UNSW dataset. Erfani [9] combined a DBN (Deep belief Network) and SVM. The first layer of the proposed architecture used DBN to extract features, while the second SVM layer was trained on the features extracted by the DBN network. The results were powerful and fitting high-dimensional domains. Dong [10] proposed a deep learning-based intrusion detection model, AEAlexJNet, which uses Auto-Encoder AlexNet neural network. The experimental results of the intrusion detection data set KDD99 show that the accuracy of the AE-AlexNet model is 94.32%. Dutt [11] imitates the adaptive immune system by taking into consideration the activation of the T-cells and the B-cells. It captures relevant features from header and payload portions for effective detection of intrusion. Experiments have been conducted on both the real-time network traffic and the standard datasets KDD99 and UNSW-NB15 for intrusion detection. The SMAD model yields is 96.04% true positive rate and around 97% true positive rate using real-time traffic and standard data sets. Hanselmann [12] used a neural network architecture for detecting intrusions on the controller area network (CAN), and introduced an unsupervised learning approach called CANet, evaluated on real and synthetic CAN data. A comparison with previous machine learning based methods shows that CANet outperforms them by a significant margin. CNN (Convolutional Neural Network), most commonly applied to analyzing visual imagery, was also used in intrusion detection. Khan [13] introduced an improved CNN on KDD99 dataset. The results showed that CNN can greatly improve the accuracy. Another deep learning algorithm was used by Javaid [14] to construct an NIDS called Self-taught Learning (STL). Experimental results were very encouraging and revealed that this architecture is as good as the previous models. Vijayanand [15] proposed a wrapper-based approach using the modified whale optimization algorithm (WOA) and used a method in which the genetic algorithm operators were combined with the WOA. Using a support vector machine (SVM), then, the authors identified the types of intrusions based on the selected features. Sydney [16] proposes a Feed-Forward Deep Neural Network (FFDNN) wireless IDS system using a Wrapper Based Feature Extraction Unit (WFEU) and compare it to standard machine learning (ML) algorithms that include Random Forest (RF), Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT) and k-Nearest Neighbor (kNN). The experimental studies include binary and multiclass types of attacks. The solution was tested on UNSW-NB15 and the AWID dataset with 87.10% and 77.16% accuracy for the binary and multiclass classification. Shapoorifard [17] used an improved version of K-Nearest Neighbors (KNN) classifier. The author combines K-MEANS clustering and KNN classification and claims that engaging the farthest neighbor enhances the accuracy rate and detection rate and reduces false alarm rate. Tama [18] designed an anomaly-based IDS, with two-step classifier. The TSE-IDS model select features using a hybrid method and collect specific feature representations. The results on the NSL-KDD and UNSW-NB15

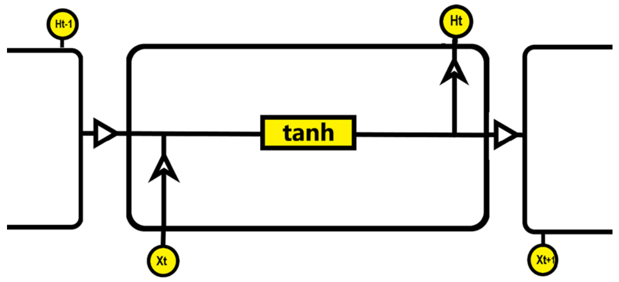


Fig. 2 Recurrent neural network architecture

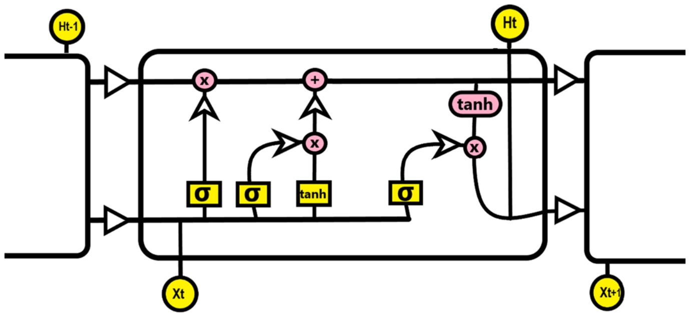


Fig. 3 Long short-term memory (LSTM) architecture

datasets improved the detection rates. Li [19] proposed an AE-IDS (Auto-Encoder Intrusion Detection System) based on random forest algorithm. This method constructs the training set with feature selection and feature grouping. After training, the model can predict the results with auto-encoder. Su [20] combines a bat model and BLSTM (Bidirectional Long Short-term memory) and attention mechanism. Attention mechanism is used to screen the network flow vector composed of packet vectors generated by the BLSTM model, which can obtain the key features for network traffic classification. To the best of our knowledge, nobody compared a linear (Mutual Information) and nonlinear (Principal Component Analysis) dimension reduction algorithm, on both Binary and multiclass classification using LSTM. Shen [21] proposed an ensemble method, combining the extreme learning machine (ELM) as a base classifier and a pruning method based on the Bat Algorithm (BA) as an optimizer, resulting an accuracy of 98.94%. Khan [22] conceived an intrusion detection system, based on convolutional neural network algorithm. The entire network consists of three hidden layers. Each hidden layer contains a convolutional layer and a pooling layer.

Basic concept

Long short-term memory LSTM

LSTM is an extension of RNN, introduced by Hochreiter and Schmidhuber [23] in 1997, designed to avoid the long-term dependency issue, unlike RNN, LSTM can remember data for long periods. In RNN architecture (Fig. 2), hidden layers have a simple structure (e.g. single tanh layer), while the LSTM architecture is more complex, It is constituted of 4 hidden layers (Fig. 3)

The principal component of LSTM is the cell state. To add or remove information from the cell state, the gates are used to protect it, using sigmoid function (one means allows the modification, while a value of zero means denies the modification.). We can identify three different gates (Fig. 4):

- Forget gate layer (Fig. 4a): Looks at the input data, and the data received from the previously hidden layer, then decides which information LSTM is going to delete from the cell state, using a sigmoid function (One means keeps it, 0 means delete it). It is calculated as: $f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$
- Input/Update gate layer (Fig. 4b): Decides which information LSTM is going to store in the cell state. At first, input gate layer decides which information will be updated using a sigmoid function, then a Tanh layer proposes a new vector to add to the cell state. Then the LSTM update the cell state, by forgetting the information that we decided to forget, and updating it with the new vector values. It is calculated as: $i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$ and $\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_C)$
- Output Layer (Fig. 4c): decides what will be our output by executing a sigmoid function that decides which part of the cell LSTM is going to output, the result is passed through a Tanh layer (value between − 1 and 1) to output only the information we decide to pass to the next neuron. It is calculated as: $O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$ and $h_t = o_t * tanh(C_t)$

Feature selection

Having a large number of dimensions in the feature space can dramatically impact the performance of machine learning algorithms, especially in the real-time environment.

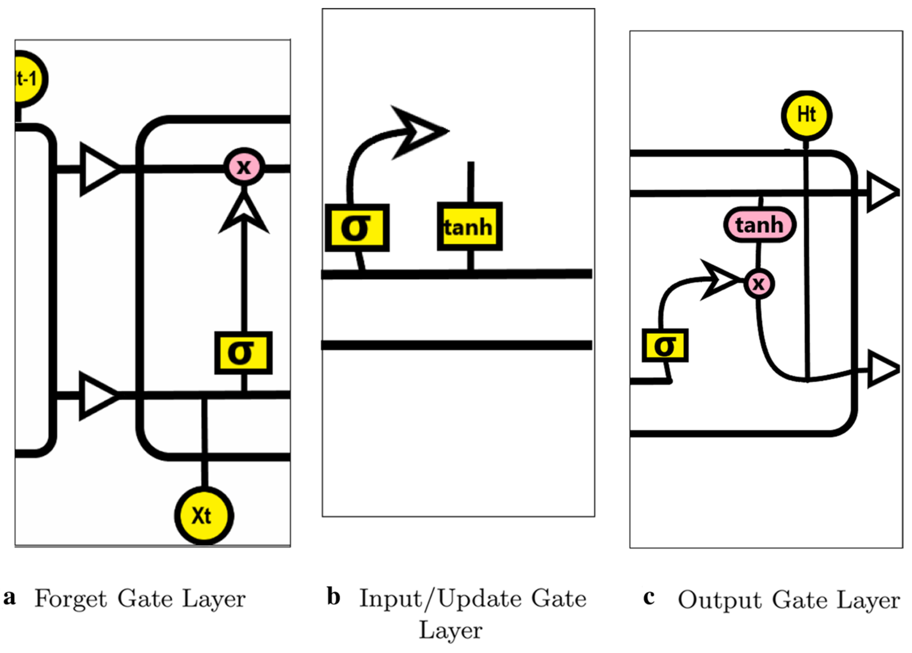


Fig. 4 LSTM layers

Therefore, many algorithms of dimensionality reduction and features selection are used on the original dataset to reduce the number of input features. This enables the machine learning algorithm to train faster, reduce the complexity of a model and make it easier to interpret.

In our research, we use two algorithms to reduce the dataset dimensionality, namely Principal Component Analysis (PCA) and Mutual Information (MI).

Principal component analysis (PCA)

Large datasets are increasingly common and are often difficult to interpret. Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, by creating new uncorrelated variables that successively maximize variance. Using PCA, we can reduce the computational costs and the error of parameter estimation by reducing the number of dimensions of the feature space by extracting a subspace that describes the data best.

Technically, after standardizing the data, PCA extracts the eigenvectors and eigenvalues from the covariance matrix (CM):

$$CM = \frac{1}{n-1}((X - x')^T(X - x'))$$

where x' is the mean vector $x' = (\frac{1}{n}) \sum_{n}^{k=1}(x_i)$. and the covariance between two features:

$$Cv_{jk} = \left(\frac{1}{n-1}\right)\sum_{n}^{i=1}(x_{ij} - x'_j)(x_{ik} - x'_k)$$

The eigenvalues are then arranged in descending order, and *k* eigenvectors corresponding to *k* eigenvalues are chosen, where *k* is the number of dimensions of the new feature subspace ($K < d$). Next, PCA builds the projection matrix W from the selected *k* eigenvectors. And finally, it transforms the original dataset X via W to obtain a k-dimensional feature subspace $Y = X * W$.

Mutual information (MI)

Unlike PCA, Mutual information calculates the statistical dependence between two variables. It measures how much information is communicated, on average, in one random variable about another and assigns a score for each feature. High MI score between two variables indicates a large reduction in uncertainty; low mutual information indicates a small reduction while zero mutual information score means the variables are independent.

Therefore, the mutual information between two discrete variables *X*| AND *Y*| denoted *I*(*X*; *Y*), is defined by Cover and Thomas [24] as:

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} = E_{P_{XY}} log \frac{P_{XY}}{P_X P_Y}$$

Here $P_X(x)$ and $P_Y(y)$ are the marginals: $P_X(x) = \sum_y P_{XY}(x,y)$.

Our approach

This section includes the dataset description, data preprocessing, calculation of the PCA variance, calculation of Mutual information scores, implementation and parameters of the proposed model.

The KDD99 dataset

Although KDD99 dataset [25] is more than 19 years old, it is still widely used in academic research. The raw training data were processed into about five million connection records. A connection is a sequence of TCP packets starting and ending at some well-defined times. The dataset contains 53 features, and falls into 4 attacks categories (DoS, Probe, U2R, and R2L) divided into 22 different attacks (Table 1):

- Denial of service attack (DoS): is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed.
- User to root attack (U2R): is an attack where hackers exploit some vulnerabilities to gain root access to the system, providing him unauthorized access to the local superuser.
- Remote to local attack (R2L): occurs when the attacker finds vulnerable points in a computer or network security software to gain access to the machine or the system. The main goal of this attack is to explore or steal data illegally, introduce viruses or cause damage to the victim. Probing Attack: Also called Scanning/Discovery, which is the first step of an attack; probing is made to gather information on the targeted system. The network is scanned for known vulnerabilities in infrastructure software, as well as unknown vulnerabilities in the custom code developed for the specific target application.

To improve the quality of raw data, data preprocessing and filtering are required which increase data efficiency. Therefore, KD99 was imported as a Panda Data Frame, then, we collected statistical information about each attack type instance. As we can see (Table 2), the number of attack records is much bigger than normal records. This unbalanced data can lead our model to fail in the classification task. To overcome this problem, we used some sampling techniques that we describe in the next section.

Table 1 Categories of attacks of KDD99

Classification of attacks	Attack name
Probe	Portsweep, IPsweep, Nmap, Satan
DoS	Neptune, Smurf, Pod, Teardrop, Land, back
U2R	Bufferoverflow, LoadModule, Perl, Rootkit
R2L	Guesspassword, Ftpwrite, Imap, Phf, Multihop, Warezmaster, Warezclient

Table 2 Number of instances for each type of attack

Attack type	Number of instances
SMURF(DOS)	2,807,886
NEPTUNE(DOS)	1,072,017
Back (DOS)	2,203
POD (DOS)	264
Teardrop (DOS)	979
Buffer overflow (U2R)	30
Load module (U2R)	9
PERL (U2R)	3
Rootkit (U2R)	10
FTP write (R2L)	8
Guess password (R2L)	53
IMAP(R2L)	12
MulitHop (R2L)	7
PHF (R2L)	4
SPY (R2L)	2
Warez client (R2L)	1,02
Warez master (R2L)	20
IPSWEEP (PROBE)	12,481
NMAP (PROBE)	2,316
PORTSWEEP(PROBE)	10,413
SATAN (PROBE)	15,892
Normal	972,781

Table 3 Attack records for binary classification, before and after sampling

Record type	Before sampling	After sampling
Normal	972,781	100,000
Attacks	3,925,650	100,148

Data preprocessing

Binary classification

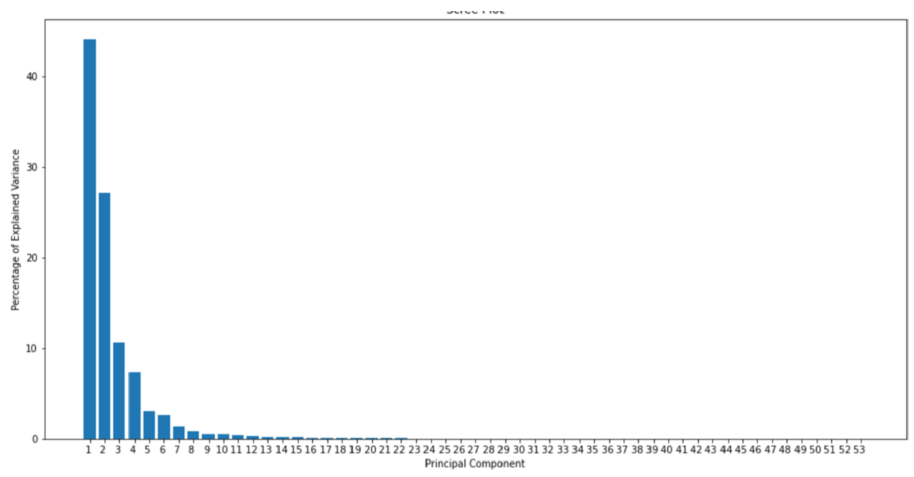
Binary classification is the task of labeling output into two groups. In our case, our binary classifies should have the ability to decide whether a given record is an attack or not. To do that, we group label into two categories: Normal and Attack. Furthermore, to overcome unbalanced data issues, we used random sampling. Hence, we choose 100,000 records for the normal category and 100,148 records for attacks category (Table 3).

Multiclass classification

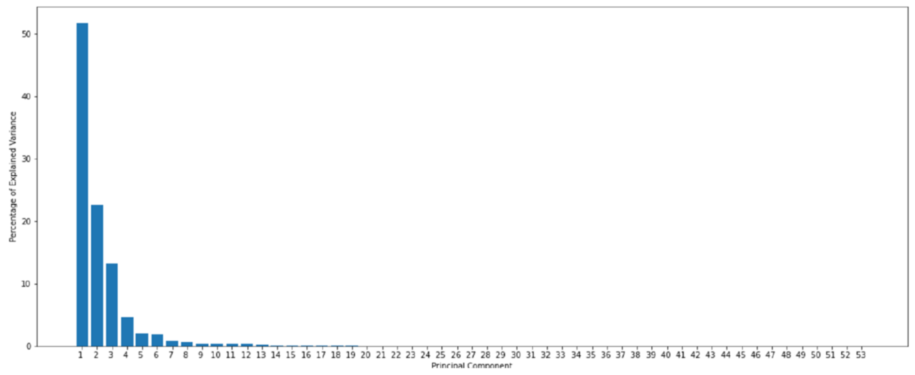
Unlike binary classification, which classifies instances into two classes, multiclass classification outputs into three or more classes. Due to the unbalanced data problem, we grouped attacks into three categories of attacks which are: Normal, DoS and all the

Table 4 Attack records for multiclass classification, before and after sampling

Record type	Before sampling	After sampling
Normal	972,781	120,000
DoS	3,925,650	100,000
R2L	1,114,267	100,000



a Percentage of Explained Variances for binary classification



b Percentage of Explained Variances for Multiclass classification

Fig. 5 PCA calculus

others in the R2L category. Then, we sampled them in 120,000, 100,000 and 100,000 records respectively (Table 4).

PCA and mutual information

Principal component analysis (PCA)

As we mention above, PCA is a technique to reduce the dataset dimensionality.

For that, we calculate the percentage of explained variances between the 53 features for binary classification (Fig. 5a) and multiclass classification (Fig. 5b).

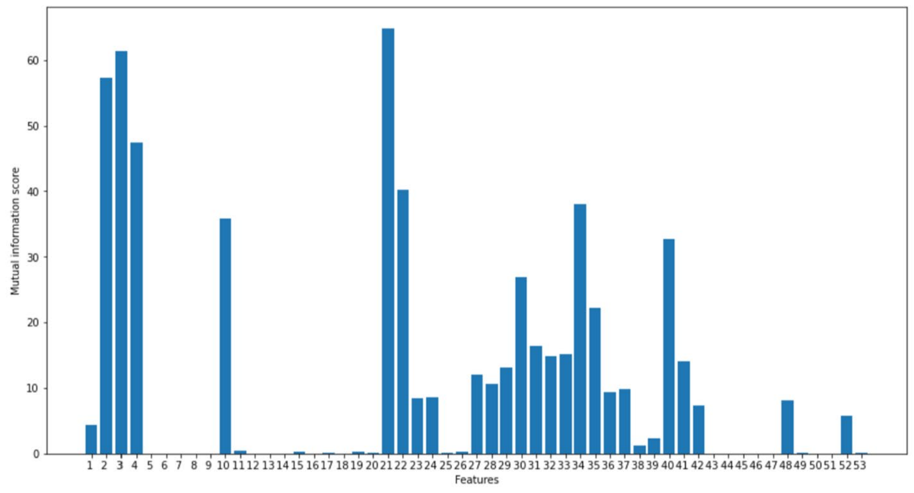
As we can see, the first two components represent more than 71% of the data for binary classification, and 74% for multiclass classification. Therefore, if we add the third component, we get more than 81% for binary classification and 87% for multiclass classification. As a result, we tested our model using a reduced dataset of two and three components.

Mutual information (MI)

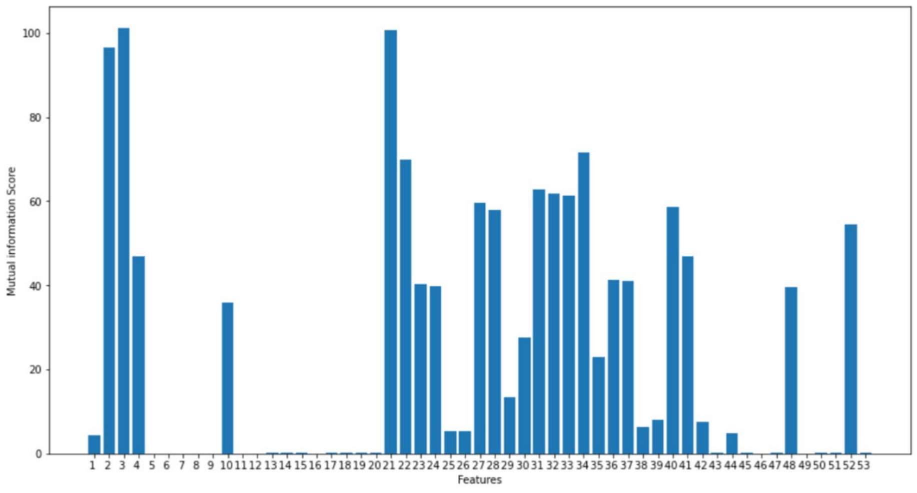
We indicated earlier that Mutual information calculates the statistical dependence between two variables. Thus, a score is assigned to each feature, showing how much the latter impacts the result.

For that, we calculated the mutual information scores, for binary (Fig. 6a), and multiclass (Fig. 6b).

For binary classification, four features have the best mutual information scores, namely feature 21, feature 3, feature 2 and feature 4. While 10 features have relatively good scores, the other don't, which means they didn't impact the classification result.



a Mutual information scores for binary classification



b Mutual information scores for multiclass classification

Fig. 6 Mutual information calculus

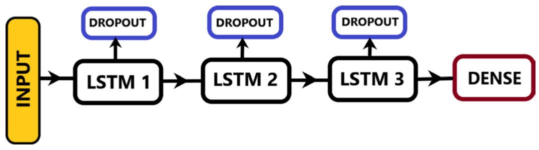


Fig. 7 The proposed LSTM architecture

Table 5 Table of parameters of the proposed model

Parameter	Binary	Multiclass
Activation function	Sigmoid	Softmax
Loss function	Binary crossentropy	Sparse categorical crossentropy
Optimizer	Adam	Adam
Learning rate	0.002	0.002
Epsilon	1e-08	1e-08
Schedule decay	0.004	0.004
Epochs	50	50
Dropout	0.1	0.1

Thus, we tested our model with 04 and 10 features scores. Concurrently, we tested with the same number of features for multiclass classification.

Implementation and evaluation metrics

We have implemented our model on Google Colab, using Keras library (Which is an open-source neural network library written in Python). In Google Colab, we split the preprocessed Dataset into 60% for training Data, 20% for validation and 20% for testing purposes.

On the other hand, our LSTM model (Fig. 7) was configured with the parameters shown in the Table 5.

To determine the performance of the proposed algorithm, a confusion matrix was plotted. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. Therefore, confusion Table matrix allows us to calculate the true positives, true negatives, false positives and false negatives such as:

True positive (TP) is when the model classifies an attack as an attack.

True negative (TN) is When the model classifies a normal entry as normal.

False positive (FP) is when the model classifies a normal entry as an attack.

False negative (FN) is When the model classifies an attack as a normal entry.

Usually, intrusion detection systems try to reduce the false positive and false negative rate, knowing that the latter (FN) have severe consequences on information systems.

In our research, we used multiple metrics namely: Accuracy, Sensitivity (or Recall), precision and F1 Score. These metrics are calculated as:

$$Accuracy = (TP + TN)/AllPredictions$$

$$Sensitivity(Recall) = TP/(FN + TP)$$

$$Precision = TP/(TP + FP)$$

$$F1Score = 2 * (Precision * Sensitivity)/(Precision + Sensitivity)$$

Experimental results

In this research, we gathered binary classification results and multi-classification(3-class) results. In both groups of classification, we used PCA and Mutual information for dimensionality reduction, and we applied LSTM as a classification algorithm.

Figure 8 summarize the performance values of each model, in term of training accuracy, testing accuracy, sensitivity (recall), precision and F1 scores.

For binary classification

In this experiment, we noticed that LSTM-PCA provided the best results, especially using 02 components. This may be due to that PCA had low noise sensitivity, remove correlated features and smaller dimensions help the classifier to learn easily. We can also notify that Mutual information is better with 4 features than 10 features and that LSTM performs well in the binary classification, but for multi-classification is much less efficient, it may be due that the data set is quite noisy. The confusion matrices of binary classification models are presented in Fig. 9.

For multiclass classification

Same as binary classification, we noticed that LSTM-PCA produced the best results, again, using only 02 dimensions.

The confusion matrices of the multiclass models are shown in Fig. 10.

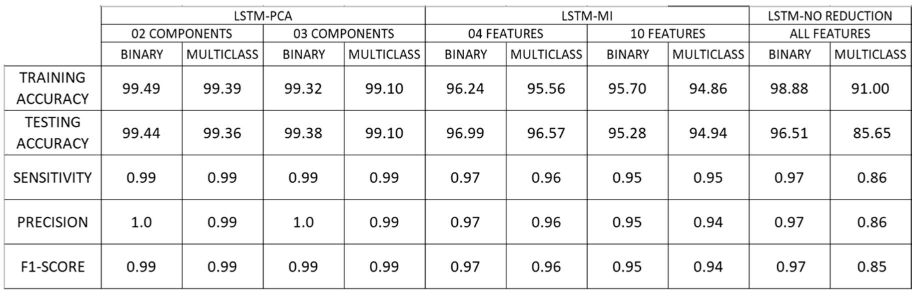
Processing time

Another important metric is the time processing. Processing times tell us how long we can expect it will take us to process a request.

In training 04 features models, LSTM-PCA takes 1397.74 seconds and 876.68 seconds in multiclass and binary classification respectively. While, in LSTM-MI models, 1348.88 seconds and 732.59 seconds have been noticed in multiclass and binary classification respectively.

On the other hand, training 10 features models takes 3,499.26 seconds, 2,209.40 seconds for LSTM-PCA and 3,166.27 seconds, 2,109.26 seconds for LSTM-MI, in multiclass and binary classification respectively.

Finally, using all the dataset features without any dimensionality reduction, takes 11,629.69 seconds and 18,802.28 seconds for multiclass and binary classification.



	LSTM-PCA				LSTM-MI				LSTM-NO REDUCTION	
	02 COMPONENTS		03 COMPONENTS		04 FEATURES		10 FEATURES		ALL FEATURES	
	BINARY	MULTICLASS	BINARY	MULTICLASS	BINARY	MULTICLASS	BINARY	MULTICLASS	BINARY	MULTICLASS
TRAINING ACCURACY	99.49	99.39	99.32	99.10	96.24	95.56	95.70	94.86	98.88	91.00
TESTING ACCURACY	99.44	99.36	99.38	99.10	96.99	96.57	95.28	94.94	96.51	85.65
SENSITIVITY	0.99	0.99	0.99	0.99	0.97	0.96	0.95	0.95	0.97	0.86
PRECISION	1.0	0.99	1.0	0.99	0.97	0.96	0.95	0.94	0.97	0.86
F1-SCORE	0.99	0.99	0.99	0.99	0.97	0.96	0.95	0.94	0.97	0.85

Fig. 8 Classification performance for LSTM, LTSM-PCA and LSTM-MI

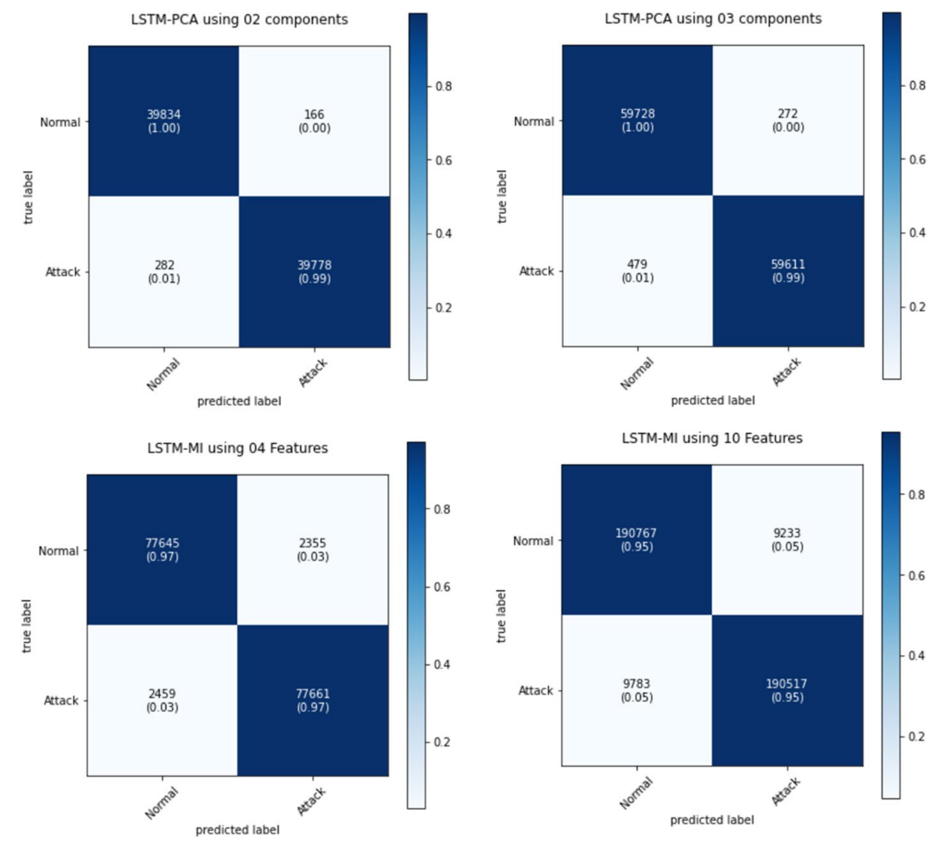


Fig. 9 Confusion matrices of binary classification models

We can remark that adding more feature increases the processing time. The training stage takes more than 15 times longer using all features compared to the models using two features. Allowing an optimizing efficiency in the training process, which saves resources.

Discussion

To improve the analysis of the experimental results, we compared the results with those of previous studies (Table 6).

Comparing multiple algorithm in such way is for reference purpose only. Intrusion detection systems vary in their classification results, as a consequence it is complex for a model to perform well in every environment. Also, as a result of data preprocessing approaches, and interpretation process, the conclusions may differ. However, it can be observed that our model (LSTM-PCA), using two features, had the best performance. It achieved the most superior accuracy and sensitivity rates.

In addition, the author of [22] used all the KDD99 features, causing a longer time for training and testing, which make this model not suitable for a realtime environment.

On the other hand, our model uses only two features. Therefore, it is the easiest to pre-process and requiring less time to train. Thus, it makes our model well suitable for large-scale and high-dimensional domains.

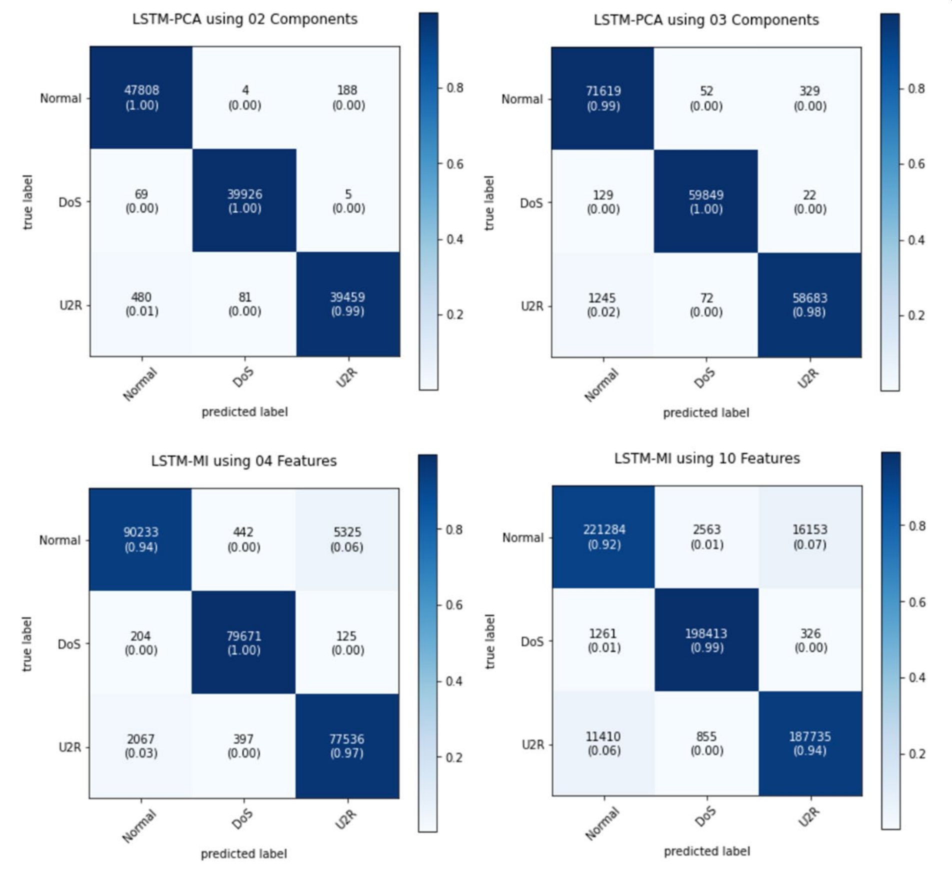


Fig. 10 Confusion matrices of multiclass classification models

Table 6 Performance comparison

Model	Accuracy	Sensitivity (Recall)
LSTM (2015) [26]	93.72%	77.07%
LSTM-RNN (2016) [27]	96.93%	98.88%
Prunning VELM(2018) [21]	98.94%	98.37%
Improved CNN (2019) [22]	99.23%	N/A
LSTM-PCA - 02 Features (proposed)	99.49%	99.15%

The global performance of our proposed architecture is more efficient than the other deep-learning models. This again proves our architecture has a better generalization performance and robustness than the compared models.

Conclusion and future works

In this paper, through the research of intrusion detection systems and neural networks, we presented a comparison, between different models, based on Long-Short Term Memory (LSTM). Principal Component Analysis and Mutual Information was used as dimensionality reduction algorithms, Then, we analyze the performance and the time processing of these approaches.

The LSTM model is capable to learn successfully the features extracted from the dataset in the training period. This capability allows the models to distinguish effectively the normal traffic from the network attacks.

To implement the proposed architectures, we used Keras library and TensorFlow on Google Colab platform. Principal Component Analysis and Mutual Information were applied as dimensionality reduction algorithms. The intrusion detection was based on binary and multiclass classification.

We noticed that architectures based on PCA, especially with 02 components had the best outcomes, for both binary and multiclass classification, with 99.44% and 99.39% respectively.

The accuracy and sensitivity are greater than the compared approaches, which also demonstrates the efficacy of the proposed method. Moreover, using only 02 features make our model easier to train, taking less time and resources than the others.

However, in this paper, we tested with only one type of LSTM. In future works, we will investigate multiple variants of LSTM (like Peephole LSTM, Multiplicative LSTM and Weighted LSTM), in addition to other neural network algorithms and other feature selection algorithms.

Acknowledgements
Not applicable.

Authors' contributions
All authors read and approved the final manuscript.

Funding
Not applicable. This research received no specific grant from any funding agency.

Availability of data and materials
Not applicable. For any collaboration, please contact the authors.

Declarations

Ethics approval and consent to participate
The author confirms the sole responsibility for this manuscript. The author read and approved the final manuscript

Competing interests
The authors declare that they have no competing interests.

Received: 24 December 2020 Accepted: 13 April 2021
Published online: 07 May 2021

References
1. Vasilomanolakis E, Karuppayah S, Muhlh auser M, Fischer M. Taxonomy and survey of collaborative intrusion detection. ACM Comput Surv. 2015;47:55.
2. Denning DE. An intrusion-detection model. IEEE Trans Soft Eng. vol. SE-13, no. 2, pp. 222–232, Feb. 1987
3. Anand Sukumar JV, Pranav I, Neetish M, Narayanan J. Network intrusion detection using improved genetic k-means algorithm. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) ;2018. https://doi.org/10.1109/icacci.2018.8554710
4. Ma T, Wang F, Cheng J, Yu Y, and Chen X. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors; 2016, vol. 16, no. 10, p. 1701.

5. Nikolov D, Kordev I, Stefanova S. Concept for network intrusion detection system based on recurrent neural network classifier. 2018 IEEE XXVII International Scientific Conference Electronics-ET;2018. https://doi.org/10.1109/et.2018.8549584
6. Jayaprakash S, Kandasamy K. (2018). Database intrusion detection system using octraplet and machine learning. 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). https://doi.org/10.1109/icicct.2018.8473029
7. Kang MJ, Kang JW. Intrusion detection system using deep neural network for in-vehicle network security. PLoS ONE. 2016;11(6):e0155781.
8. Chkirbene Z, Erbad A, Hamila R, Mohamed A, Guizani M, Hamdi M. TIDCS: A dynamic intrusion detection and classification system based feature selection. IEEE Access. 2020;. https://doi.org/10.1109/access.2020.2994931.
9. Erfani SM, Rajasegarar S, Karunasekera S, Leckie C. Highdimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognit. 2016;58:121–34.
10. Dong Y, Wang R, He J. Real-Time Network Intrusion Detection System Based on Deep Learning. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS) ;2019. https://doi.org/10.1109/icsess47205.2019.9040718
11. Dutt I, Borah S, Maitra IK. Immune system based intrusion detection system (IS-IDS): A proposed. IEEE Access. 2020;8:34929–41. https://doi.org/10.1109/access.2020.2973608.
12. Hanselmann M, Strauss T, Dormann K, Ulmer H. CANet: An unsupervised intrusion detection system for high dimensional CAN bus data. IEEE Access. 2020;8:58194–205. https://doi.org/10.1109/access.2020.2982544.
13. Khan RU, Zhang X, Alazab M, Kumar R. An Improved Convolutional Neural Network Model for Intrusion Detection in Networks. 2019 Cybersecurity and Cyberforensics Conference (CCC); 2019. https://doi.org/10.1109/ccc.2019.000-6
14. Javaid A, Niyaz Q, Sun W, Alam M . A deep learning approach for network intrusion detection system, in Proc. 9th EAI Int. Conf. BioInspired Inf. Commun. Technol. (BIONETICS), New York, NY, USA; 2015, vol. 35, pp. 21–26.
15. Vijayanand R, Devaraj D. A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. IEEE Access. 2020; https://doi.org/10.1109/access.2020.2978035.
16. Kasongo SM, Sun Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput Secur. 2020;. https://doi.org/10.1016/j.cose.2020.101752.
17. Shapoorifard H, Shamsinejad P. Intrusion detection using a novel hybrid method incorporating an improved KNN. Int J Comput Appl. 2017;173(1):5–9.
18. Tama BA, Comuzzi M, Rhee KH. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Access. 2019;7:94497–507.
19. Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur. 2020;. https://doi.org/10.1016/j.cose.2020.101851.
20. Su T, Sun H, Zhu J, Wang S, Li Y. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access. 2020;8:29575–85. https://doi.org/10.1109/access.2020.2972627.
21. Shen Y, Zheng K, Wu C, Zhang M, Niu X, Yang Y. An ensemble method based on selection using bat algorithm for intrusion detection. Comput J. 2018;61(4):526–38.
22. Khan RU, Zhang X, Alazab M, Kumar R (2019). IEEE 2019 Cybersecurity and Cyberforensics Conference (CCC) - Melbourne, Australia (2019.5.8-2019.5.9) 2019 Cybersecurity and Cyberforensics Conference (CCC) - An Improved Convolutional Neural Network Model for Intrusion Detection in Networks. 74–77.https://doi.org/10.1109/CCC.2019.000-6
23. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
24. Cover TM, Thomas JA. Information theory and statistics. In: Elements of information theory. 2nd edn. Wiley; 2005. https://doi.org/10.1002/047174882X.ch11
25. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
26. Staudemeyer RC. 1 Applying long short-term memory recurrent neural networks to intrusion detection. South Afr Comput J. 2015;56(1):136–54.
27. Kim J, Kim J, Thu HLT, and Kim H. Long short term memory recurrent neural network classifier for intrusion detection, In Proc. Int. Conf. Platform Technol. Service (PlatCon); 2016, pp. 1–5.