Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.034047
Article

Tech Science Press

An Improved LSTM-PCA Ensemble Classifier for SQL Injection and XSS Attack Detection

Deris Stiawan[1], Ali Bardadi[1], Nurul Afifah[1], Lisa Melinda[1], Ahmad Heryanto[1], Tri Wanda Septian[1], Mohd Yazid Idris[2], Imam Much Ibnu Subroto[3], Lukman[4] and Rahmat Budiarto[5,*]

[1]Universitas Sriwijaya, Palembang, 30319, Indonesia
[2]Universiti Teknologi Malaysia, Johor, 81310, Malaysia
[3]Universitas Islam Sultan Agung, Semarang, 50112, Indonesia
[4]Directorat General of Higher Education Research and Technology, Jakarta, 10270, Indonesia
[5]Albaha University, Alaqiq, 65779-7738, Saudi Arabia
*Corresponding Author: Rahmat Budiarto. Email: rahmat@bu.edu.sa
Received: 05 July 2022; Accepted: 08 December 2022

Abstract: The Repository Mahasiswa (RAMA) is a national repository of research reports in the form of final assignments, student projects, theses, dissertations, and research reports of lecturers or researchers that have not yet been published in journals, conferences, or integrated books from the scientific repository of universities and research institutes in Indonesia. The increasing popularity of the RAMA Repository leads to security issues, including the two most widespread, vulnerable attacks i.e., Structured Query Language (SQL) injection and cross-site scripting (XSS) attacks. An attacker gaining access to data and performing unauthorized data modifications is extremely dangerous. This paper aims to provide an attack detection system for securing the repository portal from the abovementioned attacks. The proposed system combines a Long Short–Term Memory and Principal Component Analysis (LSTM-PCA) model as a classifier. This model can effectively solve the vanishing gradient problem caused by excessive positive samples. The experiment results show that the proposed system achieves an accuracy of 96.85% using an 80%:20% ratio of training data and testing data. The rationale for this best achievement is that the LSTM's Forget Gate works very well as the PCA supplies only selected features that are significantly relevant to the attacks' patterns. The Forget Gate in LSTM is responsible for deciding which information should be kept for computing the cell state and which one is not relevant and can be discarded. In addition, the LSTM's Input Gate assists in finding out crucial information and stores specific relevant data in the memory.

Keywords: LSTM; PCA; ensemble classifier; SQL injection; XSS

1 Introduction

SQL injection (SQLi) and cross-site scripting (XSS) are vulnerable attacks that exist in web applications [1,2]. Li et al. [3] include the SQLi attack among the top five web application security threads. It becomes

increasingly dangerous when an attacker gains access to the data and performs unauthorized data modifications [4,5]. SQLi is performed by inserting SQL commands into web forms, domain names, or page queries and ultimately tricks the server into running malicious SQL commands, causing significant damage to websites and users [6]. XSS occurs when a malicious web code is sent, typically in the form of a script via a browser on the victim's computer. Through XSS, attackers can retrieve personal information and steal user data. XSS has been listed several times among the Top 10 web application security risks in the Open Web Application Security Project, with 37.2% of occurrences, and is still king of the hill for high-risk issues [7]. On the other hand, the Open Web Application Security Project mentioned that SQL injections were the third most serious web application security risk in 2021, with 274,000 occurrences recorded. The SQLi and XSS payloads from the attack will be stored in the database, resulting in unstable PhpMyAdmin dashboard access because they contain all the payloads.

Since the year 2019, the Directorate General of Higher Education of the Ministry of Education of the Republic of Indonesia has developed three portals for disseminating outcomes of research in Indonesia. The first is the Science and Technology Index (SINTA) which measures the academicians' and researchers' performance; the second is the "Garba Rujukan Digital" (GARUDA) which collects all publications at any journals and publishers; and the third is the RAMA, which spreads and distributes research works carried out by undergraduate and postgraduate students in Indonesia [8]. The three portals (RAMA, SINTA, and GARUDA) have been integrated into a cloud network/system with several services and applications which complete each other. As the three portals are national assets, they are undoubtedly vulnerable to hacker attacks and cyber threats. Statistics of the portals' visitors show an increasing trend yearly. The portal can also be accessed from different continents. Millions of visitors access the SINTA portal, while RAMA's daily hits average more than 2000 visitors. Preliminary research showed that thousands of accesses were suspected of carrying out attempted attacks. Fig. 1 illustrates the complexity of the attack traffic captured from the RAMA-SINTA-GARUDA cloud network.

Previous studies have discussed the detection and prevention of SQLi via the classification of machine learning methods using a Support Vector Machine (SVM), producing an accuracy of 98% [8–10]. In the studies, the authors suggest that further research using other machine learning methods with multi-class data is required because previously employed methods used binary data in their studies. Another study has discussed SQLi detection using Multi-Layered Perceptron (MLP) and long short-term memory (LSTM) networks, both of which achieved 97% accuracy, with MLP having a bit higher accuracy compared to the LSTM [11]. The keyword recognition capabilities and special character detection using LSTM need improvement. The detection of unique characters, symbols, and numeric size differences remains unclear, which blurs the boundary between letters and characters in the LSTM detection process. Li et al. [3] discuss SQLi detection using the LSTM network and compare it with the MLP and K-Nearest Neighbors (KNN) method. The MLP and KNN methods achieved an accuracy level of 87% and 89%, respectively, and a vanishing gradient owing to an unbalanced spread of positive and negative data was observed. Meanwhile, the LSTM network produced the highest accuracy rate of 91%.

By referring to the research work carried out by Li et al. [3], this paper hypothesizes that the LSTM network can be used for SQLi and XSS attack detection on the RAMA Repository to solve the vanishing gradient problem. Thus, the main contribution of this paper is an ensemble of PCA feature selection with LSTM classifier for the development of SQLi and XSS attack detection systems. This paper also contributes towards identifying essential features that represent the appearance of the SQLi and XSS attack packets in network traffic.

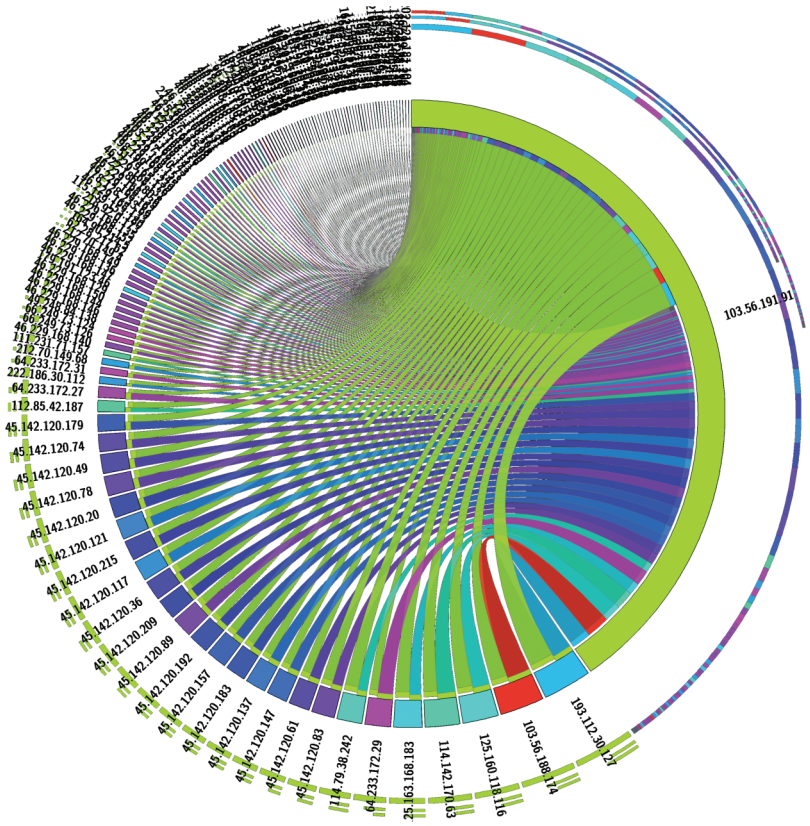

Figure 1: Real-time attack attempt traffic on the RAMA server

2 Related Works

Li et al. [3] discussed an SQL injection detection system in intelligent transportation using LSTM networks, obtaining an accuracy of 91.79%. The experiment results showed that the sample positive generation method for data collection can reduce the problem of overfitting, which is beneficial for SQL injection detection. Xie et al. [6] researched SQL injection detection based on Elastic-Pooling Convolutional Neural Network (EP-CNN) using approximately 4.48 million real-time web log data. The study concluded that the introduction of SQL injection detection based on EP-CNN automatically extracts common hidden features in SQL injection and identifies attack traffic rapidly bypassing regular SQL injection. Zhou et al. [7] introduced a new model for detecting XSS based on the Bayesian network algorithm. The study proposed XSS attack detection based on ensemble learning approaches using domain knowledge and threat intelligence. This approach achieved an accuracy rate of 96.9%, which is better than the accuracy of several other algorithms, such as SVM, Random Forest, and Decision Tree.

Guo et al. [10] have researched the detection and prevention of SQL injection using the SVM algorithm in a 5G network. The SVM classifiers classify the packets into normal or malicious packets. The calculated throughput and intrusion detection rate are higher, while the latency, energy consumption, and packet loss rate are lower, indicating that the proposed approach has achieved better QoS. Alshunaifi et al. [11] used MLP and LSTM networks for SQL injection detection. The study concluded that LSTM has excellent potential in threat intelligence detection. A malicious Uniform Resource Locator (URL) detection was proposed by Tang et al. [12], obtaining an accuracy rate of 75%. Selvaganapathy et al. [13] discussed SQL injection detection using MLP and LSTM networks. The results showed that the extracted eigenvalues effectively detect SQL injection. This method can effectively avoid the gap caused by the

filtering method. In a research work by Tang et al. [14], SQLi attack classification was carried out using a deep convolution neural network (CNN). XSS attack detection system using the decision tree algorithm with a detection accuracy of 93.70% was reported by Kascheev et al. [15]. The proposed system is superior to other algorithms, such as the Naïve Bayes classifier, SVM, and logistic regression. Akaishi et al. classified XSS attacks using machine learning with a frequency of appearance and co-occurrence [16]. In research work by Abaimov et al. [17], SQL injection and XSS detection system using code-injection detection with deep (CODDLE) was introduced. CODDLE is designed to support static methods and to analyze whether the query fully matches the existing tokens in the database with known instances. Then, numerical results are generated using the datasets that include SQL injection and XSS attacks. The results showed that CODDLE provided up to 94% accuracy. Meanwhile, Christy et al. [18] used deep learning to detect web attacks. Table 1 compares recent works on SQLi and XSS attack handling.

Table 1: Recent works on SQLi and XSS detection

Title	Method	Result
LSTM-based SQL injection detection method for intelligent transportation system [3]	LSTM	Accuracy 91.79%
An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence [7]	Ensemble learning	Accuracy 96.9%
Deep belief network-based detection and categorization of malicious URLs [12]	LSTM	Accuracy 75%
CODDLE: Code-injection detection with deep learning [17]	CODDLE	Accuracy 94%
The detecting XSS using machine learning methods [15]	Decision tree	Accuracy 93.70%
An improved LSTM network with PCA for SQL injection and XSS attack detection (Proposed)	LSTM + PCA	Precision 97% Accuracy 96.85% Recall 97% F1 Score 97% Exec time 4800 s

3 Material and Method

Fig. 2 illustrates the overall steps of the research methodology. The first step is preparing and cleaning SQL injection and XSS traffic datasets, followed by performing dynamic analysis, which includes: feature extraction on the attack traffic and normal traffic using the phpMyAdmin tool; and exporting the data from the .sql format to .csv format. After performing feature extraction, feature selection is done to obtain the best features for the classification process. Next, experiments on attack detection using LSTM, LSTM + PCA, Gated Recurrent Unit (GRU), and validation are carried out. The experiment results are then visualized.

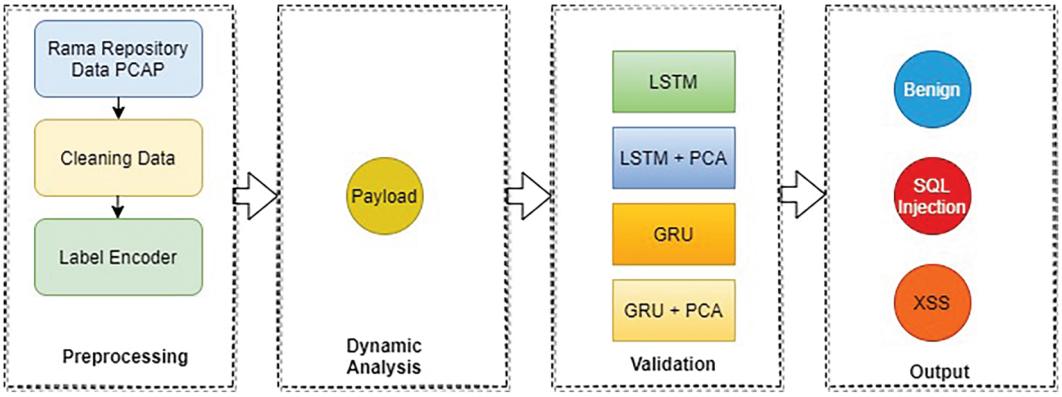

Figure 2: Research methodology

3.1 Dataset and Its Preprocessing

This paper uses the SQLi and XSS datasets from the RAMA Repository. The Rama repository handles the information database storage through data harvesting from affiliations' repositories via OAI XML. The dataset consists of the normal packets and two types of attack packets i.e., SQL injection and XSS. The labeling of attack data is done following the syntax structure of the attack packets. The total number of records in the dataset is 1,959,747, where 1,948,277 are normal packets, 11,074 are SQL injection attack packets, and 396 are XSS attack packets. However, this paper only considers 1,000,000 normal packets in the experiments. The distribution (in percentage) of the packets in the dataset is shown in Fig. 3. The dataset has been made publicly available at the Comnet Laboratory portal [19].

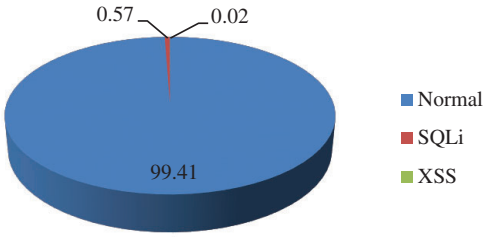

Figure 3: Distribution of the packets in the dataset

The dataset is in the form of .sql format, which will later be converted into a .csv file using MySQL tools. Preprocessing starts with first cleaning the data, followed by the label encoding process, which converts text data into numeric form. The last step in data preprocessing is data splitting, where the dataset is split into training data and testing data with four different ratios.

3.2 Dynamic Analysis of Attack Packets

During the analysis stage, the requested URL is assessed to detect a particular pattern from the record/data in a table field based on the specified string pattern and fill in the syntax/attack pattern in the value column [20].

This paper extracts the SQL tautology type from the attacker's Internet Protocol (IP) address field. The attack payloads are listed in the corresponding *requested_url* column. Table 2 shows the SQL tautology types extracted from four attackers that exist in the RAMA-SINTA-GARUDA cloud network.

Table 2: Captured SQL injection payload

The attacker's IP address	Requested URL
74.220.215.110	database record or (1, 2) = (select * from (select name_const (CHAR (111, 108, 111, 108, 111, 115, 104, 101, 114), 1), name_const (CHAR (111, 108, 111, 108, 111, 115, 104, 101, 114), 1))a)–and 1 = 1
103.56.189.55	/search/all/?q=%28SELECT%20%28CASE%20WHEN%20%289012%3D7813%29%20THEN%20%27sd%27%20ELSE%20%28SELECT%207813%20UNION%20SELECT%203697%29%20END%29%29
140.213.32.67	/search/all?q=Studi%20Tentang%20Gaya%20Pada%20Interior%20Masjid%20Kampus%20Universitas%20Gadjah%20Mada%20Yogyakarta'%2bselect%20load_file('%5c%5c%5c%5caesdlur6qzajy489pn4texv2otunie659t0gq4f.burpcollaborator.net%5c%5cffy'))%2b'
125.161.137.239	/afiliasi/all?q=syscolumns%20WHERE%202%3e3%3bDECLARE%2f**%2f%40x%2f**%2fchar(9)%3bSET%2f**%2f%40x%3dchar(48)%2bchar(58)%2bchar(48)%2bchar(58)%2bchar(50)%2bchar(53)%3bWAITFOR%2f**%2fDELAY%2f**%2f%40x–

The attacker's IP address 74.220.215.110 attempts an SQL injection attack via attack payloads randomly generated by the SQL_map automatic tool, and the payload uses the *name_const* parameter, wherein the parameter selects certain CHAR characters. The attacker's IP address 103.56.189.55 selects the UNION operation on a specific ID number present in the search path website query parameters (/search/all/?q=), and the payload in *URL-encoded*, so that the browser can execute the injected SQL command.

The attacker's IP address 140.213.32.67 performs an SQL injection attack with the option *load_file* and input URL of burp collaborator (*aesdlur6qzajy489pn4texv2otunie659t0gq4f.burpcollaborator.net*); the attacker will get a response from the target in the form of the server IP information and DNS IP server information. The attacker's IP address 125.161.137.239 automates SQL injection attacks using the SQLmap tool by selecting the highest level and the highest risk option, which can be seen from the generated payload.

Table 3 shows the XSS payload of two attackers in the network. The attacker's IP Address 140.213.32.67 inputs XSS payload in the HTML-script format; this script is known to trigger pop-up XSS alerts on specific browsers.

An attacker's IP Address 202.67.43.1 tries to input XSS payload using the JavaScript code format, but the mod-security filter detected it on the target website; thus, the payload gets deleted. It can be seen from the removed string that the attacker intended to trigger a pop-up XSS alert on the browser. If the website has such the vulnerability, the attacker can change the payload and steal cookies from a specific target user.

3.3 Feature Extraction

The payload data (packet) structure consists of fields. The fields are considered as attributes and then as features of the packet. At this feature extraction stage, firstly, the .sql data is converted into .csv format. The features extraction process is performed using the XAMPP tool that connects to the phpMyAdmin localhost, where the .sql format file is exported to .csv format. The extracted features/attributes are displayed in Fig. 4.

Table 3: Captured XSS payload

The attacker's IP address	Requested URL
140.213.32.67	/afiliasi/all?q=mercualert('XSS'
202.67.43.1	/afiliasi/all/?q=unsri[removed]alert(45)[removed]/search/all/?q=query"<xspear xss=removed>

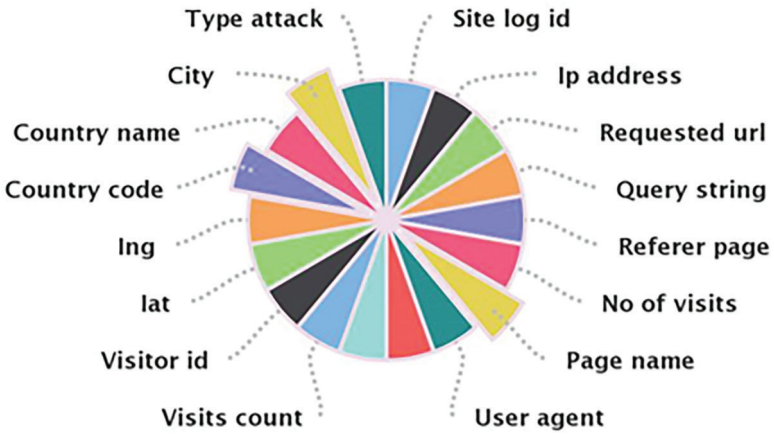

Figure 4: Attributes of the dataset

3.4 PCA

A colossal dataset has high data dimensionality. To interpret efficiently a colossal dataset, this paper needs to reduce the dataset dimensionality while at the same time preserving the information in the dataset. Many approaches have been introduced for reducing the dimension of colossal datasets. The PCA is one of the most widely used approaches due to its efficiency in obtaining the relevant and significant features of the dataset [21]. Therefore, this paper chooses PCA as the data dimensionality reduction method. The PCA finds new variables i.e., the principal components, which are uncorrelated and ordered so that the first few retain most of the variation that exists in all of the original variables. Mathematically, the PCA attempts to solve an eigenvalue/eigenvector problem. The PCA equations are shown in (1) to (4).

$$\overrightarrow{a_1} = z_{11}\overrightarrow{x_1} + z_{12}\overrightarrow{x_2} + \cdots + z_{1p}\overrightarrow{x_p} = \overrightarrow{z_1}^T X \tag{1}$$

$$\overrightarrow{a_2} = z_{21}\overrightarrow{x_1} + z_{22}\overrightarrow{x_2} + \cdots + z_{2p}\overrightarrow{x_p} = \overrightarrow{z_2}^T X \tag{2}$$

$$\overrightarrow{a_p} = z_{p1}\overrightarrow{x_1} + z_{p2}\overrightarrow{x_2} + \cdots + z_{pp}\overrightarrow{x_p} = \overrightarrow{z_p}^T X \tag{3}$$

$$a = \begin{bmatrix} \overrightarrow{a_1} \\ \overrightarrow{a_2} \\ \cdots \\ \overrightarrow{a_p} \end{bmatrix} = \begin{bmatrix} \overrightarrow{z_1}^T X \\ \overrightarrow{z_2}^T X \\ \cdots \\ \overrightarrow{z_p}^T X \end{bmatrix} = \begin{bmatrix} \overrightarrow{z_1}^T \\ \overrightarrow{z_2}^T \\ \cdots \\ \overrightarrow{z_p}^T \end{bmatrix} X = \boldsymbol{A^T X} \tag{4}$$

This paper considers the PCA method as an unsupervised learning mechanism with the following six steps:

1. Consider the whole dataset with $(p + 1)$ dimensions and disregard the labels, so the new dataset has p dimensional.
2. For every dimension of the whole dataset, compute the mean value.
3. For the whole dataset, compute its covariance matrix.
4. Compute eigenvectors along with their corresponding eigenvalues.
5. Arrange the eigenvectors, reduce the number of eigenvalues and choose corresponding k eigenvectors with the largest eigenvalues to form a $(p \times k)$ dimensional matrix A.
6. Transform the sample matrix X onto the new subspace using the $(p \times k)$ eigenvector matrix.

3.5 Validation

To measure the effectiveness of the feature selection method discussed in Section 3.4, this paper performs a validation process using two classifier algorithms i.e., LSTM and GRU. Furthermore, this paper proposes an ensemble of PCA feature selection algorithm with an LSTM classifier algorithm and an ensemble of PCA feature selection algorithm with a GRU classifier algorithm, described in the following sub-sections. The condition of the dataset that is less varied causes a vanishing gradient condition. The Recurrent Neural Network (RNN), LSTM, and GRU methods are solutions that can solve the problem [22,23].

3.5.1 LSTM Network

Compared with recurrent neural networks (RNNs), LSTM networks can learn long-term dependencies more efficiently [20]. In LSTM networks, each traditional node in the hidden layer is replaced with a memory cell. This learning capability is the primary difference from standard RNNs [22]. The memory cell is the most essential structure that overcomes the shortcomings of ordinary RNNs in the learning process. LSTM networks can effectively solve the problem of the vanishing gradient caused by excessive positive samples. The proposed LSTM-PCA architecture is shown in Fig. 5. It can be seen that instead of using the original dataset, the reduced-dimensional dataset obtained by PCA as a feature selection algorithm is inputted into the LSTM engine.

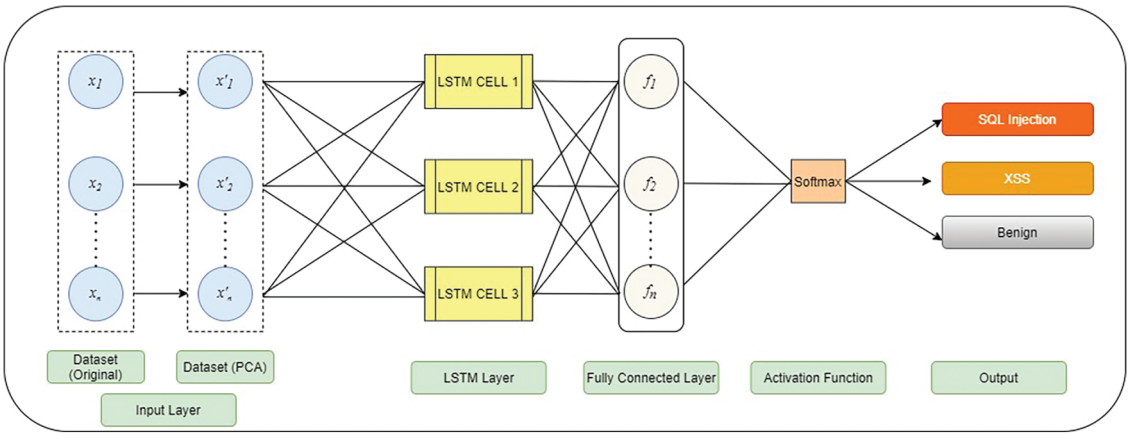

Figure 5: Proposed architecture

The memory cell comprises four main elements i.e., a neuron with a self-recurrent connection, an input gate, forget gate, and an output gate [22,24]. The Input Gate of the LSTM updates the cell state and decides which information is important and which is not. The Forget Gate in LSTM is responsible for deciding which information should be kept for computing the cell state and which one is not relevant and can be discarded.

The output gate determines the value of the next hidden state. The calculations for each LSTM update are shown in (5)–(8).

$$C_t = f_t \theta C_{t-1} + i_t \theta g_t \tag{5}$$

$$f_{(t)} = \sigma\left(W_{fx} x_t + W_{fh} h_{t-1} + b_f\right) \tag{6}$$

$$i_{(t)} = \sigma\left(W_{ix} x_t + W_{ix} h_{t-1} + b_i\right) \tag{7}$$

$$g_{(t)} = tanh\left(W_{gx} x_t + W_{gh} h_{t-1} + b_g\right) \tag{8}$$

where,

C_t: internal memory

C_{t-1}: previous internal memory

$f_{(t)}$: forget gate

$i_{(t)}$: input gate

$g_{(t)}$: hidden state candidate

W_{fx}: weight of forget gate x

x_t: input for time step t

b: bias

$tanh$: activation function

σ : sigmoid function

3.5.2 Gated Recurrent Unit (GRU)

This paper also proposes the use of GRU as a classifier engine. The GRU component consists of two gates: the reset and update gates. The GRU has no internal memory and does not have an output gate like the LSTM network [21]. The advantage of GRU is that it is computationally more straightforward and effective enough to avoid gradient vanishing problems [24]. The input and forget gates are combined as update gates, while reset gates are applied directly to the previous hidden state, so GRU has only two gates i.e., reset gate r_t dan update gate z_t. Eqs. (9)–(12) represent the GRU model:

$$z_t = \sigma(W_z \,.\, [h_{t-1},\ \ x_t]) \tag{9}$$

$$r_t = \sigma(W_r \,.\, [h_{t-1},\ \ x_t]) \tag{10}$$

$$h_{(t)} = \left(1 - u_{(t)} * h_{t-1} + u_{(t)} * \tilde{h}_{(t)}\right) \tag{11}$$

$$\tilde{h}_{(t)} = tanh\left(W \,.\left[r_{(t)} *\ h_{t-1}\right] + W x_t\right) \tag{12}$$

where,

W: weight

W_z: update gate weight

W_r: reset gate weight

$u_{(t)}$: update gate

$r_{(t)}$: reset gate

x_t: input for time step t

$h_{(t)}$: hidden candidate

$\tilde{h}_{(t)}$: hidden state candidate

h_{t-1}: previous hidden state

tanh: activation function

σ : sigmoid function

The GRU is another standard solution to address the vanishing gradient problem in RNNs, which is identified as a significant shortcoming of RNNs. GRU can be viewed as a variation of the LSTM network but with a more straightforward structure. The standard GRU is shown in Fig. 6. Similar to LSTM-PCA, the GRU is also in an ensemble with the PCA.

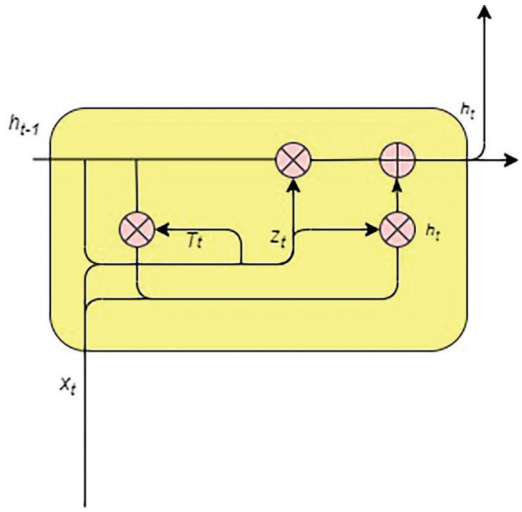

Figure 6: Gated recurrent unit

3.6 Performance Evaluation

A confusion matrix is implemented to measure the performance of a system. The system selects a model with a minimal loss value. The best model is evaluated based on the testing data. Subsequently, the system calculates and measures the Accuracy, Precision, Recall, and F1 score at the end of the evaluation process. This paper employs a multiclass system with three output classes, namely 0, for normal traffic, 1, for SQLi attack traffic, and 2, for XSS attack traffic. The equations for calculating the Accuracy, Precision, Recall, and F1 score are determined by (13) to (16) [25].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{16}$$

where *TP* is the number of true positives, *TN* is the number of true negatives, *FP* is the number of false positives, and *FN* is the number of false negatives.

4 Result and Discussion

4.1 Experiment Set up

Hardware and software specifications for the experiments are shown in Tables 4 and 5, respectively.

Table 4: Hardware specification

Device	OS	Specification
Asus A442U	Windows 10	Intel® Core™ i5-7200 CPU @ 2.500 GHz 4 GB RAM

Table 5: Software specification

System	Tool/Framework	Description
Feature extraction	XAMPP phpMyAdmin	Version 3.2.4
Validation	Python	Version 3.7

The PCA for feature selection is implemented in Python's Scikit-Learn library. The first step is normalizing the data which is followed by data splitting for training and testing. This paper needs only three lines of code to perform PCA using Python's Scikit-Learn library. The experiments are repeated five times for each dataset ratio and the average is taken as the final result. The selected features are then used by the LSTM model to classify the attack data traffic. For comparison, this paper also implemented the GRU model as a classifier.

4.2 Experimental Results

In this section, the results of experiments on the validation of the proposed system performances are highlighted. Firstly, the results of the confusion matrix are depicted in Fig. 7.

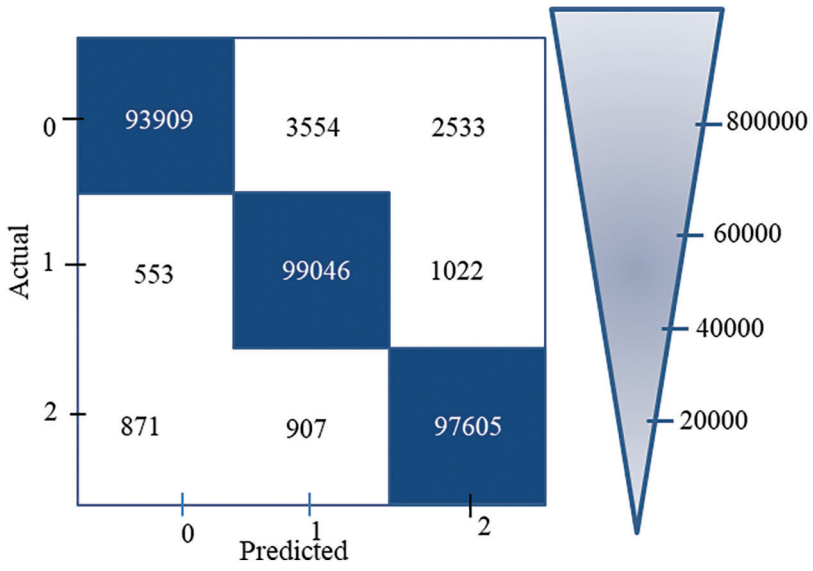

Figure 7: Confusion matrix

The confusion matrix in Fig. 7 represents the best experiment result. It is achieved when the 80%:20% ratio of the training dataset and the testing dataset is used. Tiny false negative (FN) values of 553 for normal data, 3554 for SQL injection, and 1022 for XSS were obtained. A minor FN value indicates that the proposed model is successful in classification.

Table 6 shows that LSTM–PCA is the best model classifier with an Accuracy of 0.9685 and Precision, F1 score, and Recall of 0.9700. The LSTM network has a more reliable performance than the GRU. However, GRU executes the training process faster compared to LSTM. Meanwhile, LSTM–PCA produces the best performance in terms of the vanishing gradient problem. Moreover, PCA can speed up the training process, completing it in 4800 s while maintaining the accuracy level and Precision–Recall curves as shown in Figs. 8 and 9, respectively.

Table 6: Result of the performance metrics

Train:Test	Classifier	Performance metrics				
		Precision	Accuracy	Recall	F1 score	Exec. time
40:60	LSTM	0.9125	0.8890	0.9150	0.9240	8600 s
	LSTM + PCA	0.9600	**0.9635**	0.9600	0.9600	5200 s
	GRU	0.9210	0.9015	0.8905	0.8700	8300 s
	GRU + PCA	0.9250	0.9110	0.9210	0.8990	5120 s
50:50	LSTM	0.8900	0.8900	0.8980	0.9050	8700 s
	LSTM + PCA	0.9700	**0.9663**	0.9700	0.9700	5600 s
	GRU	0.9300	0.8700	0.9050	0.9050	8780 s
	GRU + PCA	0.9370	0.9025	0.9170	0.9145	5800 s
60:40	LSTM	0.8700	0.8400	0.9200	0.8950	8630 s
	LSTM + PCA	0.9700	**0.9677**	0.9700	0.9700	5500 s
	GRU	0.8900	0.8850	0.9080	0.9050	8650 s
	GRU + PCA	0.9310	0.9030	0.9210	0.9150	5550 s
70:30	LSTM	0.9050	0.8800	0.9115	0.8900	8600 s
	LSTM + PCA	0.9700	**0.9676**	0.9700	0.9700	5600 s
	GRU	0.9250	0.9125	0.9230	0.9010	8600 s
	GRU + PCA	0.9420	0.9240	0.9310	0.9090	5700 s
80:20	LSTM	0.9530	0.9050	0.9050	0.9120	8550 s
	LSTM + PCA	**0.9700**	**0.9685**	**0.9700**	**0.9700**	**4800 s**
	GRU	0.9200	0.8990	0.9120	0.9450	7720 s
	GRU + PCA	0.9420	0.9085	0.9195	0.9510	4880 s

Fig. 9 shows the micro-average, which represents the average of the overall Precision–Recall curve and is represented by the yellow line. The blue, green, and orange lines in Fig. 9 show the average results of Precision–Recall for classes 0, 1, and 2, respectively. It is observed that the three graphs for class 0, class 1, and class 2 represent the best performance as the Precision–Recall curves show an ideal shape. The overall analysis concludes that the more data are trained, the better the Accuracy, Precision, Recall, and F1 score values.

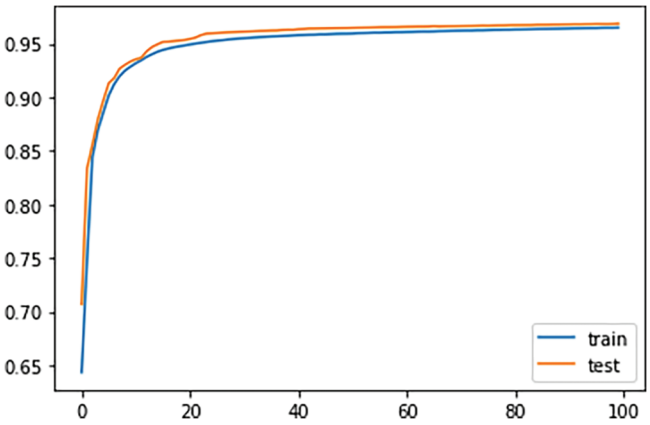

Figure 8: Precision-recall

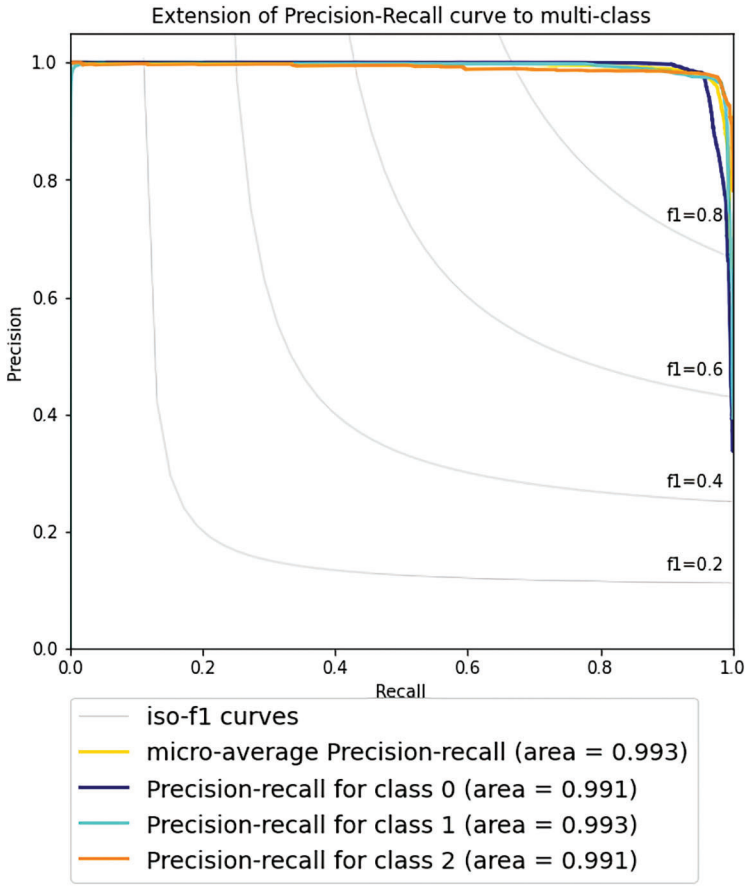

Figure 9: Precision-recall for multi-class

4.3 Discussion

The LSTM method has been successfully applied with five trials and classification on SQL Injection & XSS attacks resulting in a good performance. The 50%:50% data ratio provides an accuracy of 96.35%, the 60%:40% data ratio provides an accuracy of 96.63%, the 70%:30% data ratio provides an accuracy of 91.2%,

and the ratio of 80%:20% provides an accuracy of 96.85%. It is observed, in general, the more data training used, the more accuracy obtained.

The ensemble of the LSTM classifier and the PCA feature selection algorithm consistently always provides the best accuracy for any of the ratios of training data *vs.* testing data, as shown in Table 5. The best achievement was for the ratio of 80%:20% between training data and testing data. This result is also supported by its confusion matrix in Fig. 7. The Precision-Recall trend also supports the fact that the more significant ratio of the training data, the better accuracy obtained. This best achievement can be explained as follows. The LSTM's Forget Gate works very well as the PCA supplies selected features that are significantly relevant to the attacks' patterns. In addition, the Input Gate of the LSTM that updates the cell state and decides which information is essential or not also works better in assisting the Forget Gate in discarding unnecessary information and storing specific relevant data.

The effect of the split ratio was not significant; increasing the number of data in the training set from 40% to 70% conveyed only a tiny increase in the classification performance (Table 5). Since the detection was dedicated to multiclass classification, it is not surprising that the differences between the split ratios were not significant at small sample sizes (where the model performances were far from satisfactory anyway); however, the 70% and 80% split ratios performed better for the dataset. In these cases, the test sample was much smaller, but the performance of the models in test validations was not far from that in cross-validation.

As displayed in Fig. 3 of Section 3.1, the dataset for the experiment is imbalanced; only up to 11% of the total records are attack packets. In most circumstances, the evaluations of the performance show only up to 11% of the records presenting with features suggesting attack traffic will be finally identified as an attack, and up to 89% are not identified as attack traffic. The Precision-Recall curve is used for evaluating the performance of the proposed ensemble LSTM-PCA classifier. Predictive performance influences the Area Under the Precision-Recall Curve (AUC-PR). An ideal classifier does not make any prediction errors. Thus, it will obtain an AUC-PR of 1. The Precision-Recall curve in Fig. 9 provides a graphical representation of a classifier's performance across many thresholds rather than a single value (e.g., Accuracy, F-1 score, etc.). The values of AUC-PR for class 0, class 1, and class 2, and overall, are more significant than 0.990. It means that the proposed ensemble LSTM-PCA classifier is a good classifier even if the dataset is imbalanced.

4.4 The Attacks Detection System Implementation

This section describes the attack detection system design and implementation. Fig. 10 shows the flowchart of the detection system. The entire payload will be analyzed dynamically using MySQL procedures to come up with extracted features. Then, the ensemble of PCA and LSTM is used as a classifier to detect whether the payload contains SQL injection or XSS attacks. If the classifier recognizes that the payload has attack features, the detection system will generate an alert.

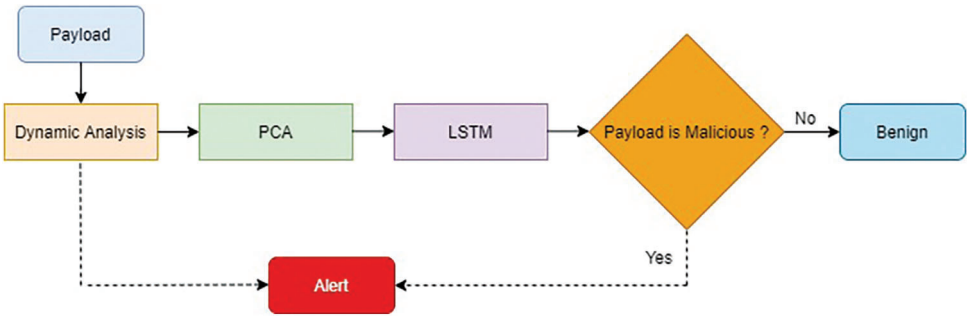

Figure 10: Attacks detection system flowchart

5 Conclusion

This paper presented the implementation of an LSTM model for classifying SQL injection and XSS attacks in the RAMA Repository. The experiment results generally show that the LSTM model is easier to train and has a more reliable performance than GRU. The GRU training process is a little bit faster, but the sheer number of missed pieces of training makes the wasted time even more remarkable. This paper also implemented PCA as a data dimensionality reduction method to speed up the computational process of SQL injection and XSS classification. The proposed LSTM–PCA-based detection system exhibits outstanding performance with an accuracy of 96.85% when classifying SQL injection and XSS attacks. The proposed classifier can also handle an imbalanced dataset.

The authors plan to develop an automatic detection system for SQL injection and XSS attacks on the RAMA-SINTA-GARUDA cloud to be implemented as a Software Defined Network (SDN) in the future. The authors also consider the use of embedded feature selection methods that include interactions of features and at the same time maintaining reasonable computational cost.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. E. Rodríguez, J. G. Torres, P. Flores and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Computer Networks*, vol. 166, pp. 106960, 2020.

[2] H. Chen, J. Chen, J. Chen, S. Yin, Y. Wu *et al.,* "An automatic vulnerability scanner for web applications," in *Proc. IEEE 19th Int. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, pp. 1519–1524, 2020.

[3] Q. Li, F. Wang, J. Wang and W. Li, "LSTM-based SQL injection detection method for intelligent transportation system," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4182–4191, 2019.

[4] M. S. Aliero, K. N. Qureshi, M. F. Pasha, I. Ghani and R. A. Yauri, "Systematic review analysis on sqlia detection and prevention approaches," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2297–2333, 2020.

[5] L. Zhang, D. Zhang, C. Wang, J. Zhao and Z. Zhang, "ART4SQLi: The ART of SQL injection vulnerability discovery," *IEEE Transaction on Reliability*, vol. 68, no. 4, pp. 1470–1489, 2019.

[6] X. Xie, C. Ren, Y. Fu, J. Xu and J. Guo, "SQL injection detection for web applications based on elastic-pooling CNN," *IEEE Access*, vol. 7, pp. 151475–151481, 2019.

[7] Y. Zhou and P. Wang, "An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence," *Computers & Security*, vol. 82, pp. 261–269, 2019.

[8] Kemendikbud, "RAMA Repository," [Online]. Available: https://rama.kemdikbud.go.id/.

[9] J. Mathew, C. K. Pang, M. Luo and W. H. Leong, "Classification of imbalanced data by oversampling in kernel space of support vector machines," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4065–4076, 2018.

[10] S. Guo, Y. Liu, R. Chen, X. Sun and X. Wang, "Improved SMOTE algorithm to deal with imbalanced activity classes in smart homes," *Neural Processing Letters*, vol. 50, no. 2, pp. 1503–1526, 2019.

[11] S. Y. Alshunaifi, S. Mishra and M. Alshehri, "Cyber-attack detection and mitigation using SVM for 5G network," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 13–28, 2022.

[12] P. Tang, W. Qiu, Z. Huang, H. Lian and G. Liu, "Detection of SQL injection based on artificial neural network," *Knowledge-Based Systems*, vol. 190, pp. 105528, 2020.

[13] S. G. Selvaganapathy, M. Nivaashini and H. P. Natarajan, "Deep belief network based detection and categorization of malicious URLs," *Information Security Journal*, vol. 27, no. 3, pp. 145–161, 2018.

[14] P. Tang, W. Qiu, Z. Huang, H. Lian and G. Liu, "SQL injection behavior mining based deep learning," In: G. Gan *et al.,* (Ed.), *ADMA 2018, Lecture Note on Artificial Intelligence*, Cham, New York, USA: Springer, vol. 11323, pp. 445–454, 2018.

[15] S. Kascheev and T. Olenchikova, "The detecting cross-site scripting (XSS) using machine learning methods," in *Proc. GloSIC*, Chelyabinsk, Russia, pp. 265–270, 2020.

[16] S. Akaishi and R. Uda, "Classification of XSS attacks by machine learning with frequency of appearance and co-occurrence," in *Proc. 53rd CISS*, Baltimore, MD, USA, pp. 1–6, 2019.

[17] S. Abaimov and G. Bianchi, "CODDLE: Code-injection detection with deep learning," *IEEE Access*, vol. 7, pp. 128617–128627, 2019.

[18] J. I. Christy Eunaicy and S. Suguna, "Web attack detection using deep learning models," *Materialstoday: Proceedings*, vol. 62, pp. 4806–4813, 2022.

[19] Comnets, "COMNETS lab dataset," [Online]. Available: https://github.com/comnetslabunsri/datasets.

[20] A. Ghafarian, "A hybrid method for detection and prevention of SQL injection attacks," in *Proc. 2017 Computing Conf.*, London, United Kingdom, pp. 833–838, 2017.

[21] M. Mateen, J. Wen, Nasrullah, S. Song and Z. Huang, "Fundus image classification using VGG-19 architecture with PCA and SVD," *Symmetry (Basel)*, vol. 11, no. 1, pp. 1–12, 2019.

[22] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, pp. 132306, 2020.

[23] X. Wang, J. Xu, W. Shi and J. Liu, "OGRU: An optimized gated recurrent unit neural network," *Journal of Physic Conference Series*, vol. 1325, no. 1, 2019. https://doi.org/10.1088/1742-6596/1325/1/012089.

[24] Ö. Yildirim, "A novel wavelet sequences based on deep bidirectional LSTM network model for ECG signal classification," *Computers in Biology and Medicine*, vol. 96, pp. 189–202, 2018.

[25] A. Rácz, D. Bajusz and K. Héberger, "Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification," *Molecules*, vol. 26, no. 4, pp. 1–16. 2021.