Proceedings of the 8th International Conference on Communication and Electronics Systems (ICCES 2023)
DVD Part Number: CFP23AWO-DVD; ISBN: 979-8-3503-9662-1

Wireless Sensor Network-Based Intrusion Detection Technique Using Deep Learning Approach of CNN-GRU

1. Anita Sagar, Post-Graduate Department of Physics, College of Commerce, Arts & Science, Patliputra - University, Patna, Bihar, India. anita.sagar75@gmail.com

2. Dr N. K Anushkannan, Associate Professor & Head, Department of ECE, Kathir College of Engineering, Coimbatore. Tamilnadu, India. anushkannan@kathir.ac.in

3. Dr G.Indumathi, Assistant Professor CSE, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamilnadu, India. gindumathi84@gmail.com

4. Dr Nikale Vasant Muralidhar, Associate Professor, Department of Physics, Ayat Shikshan Sanstha's Dada Patil Mahavidyalaya Karjat, Ahmednagar, Karjat, Maharashtra, India. vmnikale@rediffmail.com

5. Dhamotharan K A, Associate Professor, Computer Science and Engineering, Erode Sengunthar Engineering College-Autonomous, Perundurai, Erode, Tamilnadu, India. dhamuerode@gmail.com

6. P. Malini, Assistant Professor, Department of Electronics and Communication Engineering, K.Ramakrishnan College of Engineering, NH 45, Samayapuram, Trichy, Tamil Nadu, India, malinittp1995@gmail.com

***Abstract* – In a WSN, sensors are always collecting information and transmitting it to other nodes. Their primary uses are in fields like defense, smart city development, and agricultural monitoring. The WSN needs to perform at a premium for these uses. Yet, there are many potential security concerns that could compromise a WSN's performance. Any interference with the WSN could severely degrade its functionality and lead to catastrophic failures. As a result, having the ability to quickly identify and stop intrusions is crucial. The goal of this proposed is to use a GRU-CNN model for quick detection and prevention of any intrusion. After receiving the input, the proposed method uses normalization and discretization for preprocessing the data, PSO (Particle Swarm Optimization) for feature extraction, CFS for feature selection and finally training the model by CNN-GRU. To better detect intrusions in wireless sensor networks, a GRU-CNN hybrid neural network model was presented; although the CNN module is responsible for extracting the feature vector from other high-dimensional data, the GRU module is responsible for from time sequence data to extract the feature vector.**

***Keywords*—Gated Recurrent Unit (GRU), PSO (Particle Swarm Optimization), Wireless Sensor Network(WSN).**

I. INTRODUCTION

WSN development offers a promising alternative for keeping tabs on critical infrastructure, and it has been recommended for use in fields as varied as traffic monitoring, building monitoring, healthcare, and combat surveillance [1]. Every attempt to disrupt the operation of essential services has the potential to be exploited for either terrorist intentions or illegal financial gain. Network security is vital in these environments and must be implemented in a way that accounts for the constraints of these systems so that their weaknesses and vulnerabilities may be fixed. In this proposed approach to investigate the problem of incorporating intrusion detection into WSN and present a novel solution based on mobile agents to solve it. Its ease of installation and operation is a major selling advantage. Yet, creating a safe network of sensors is a major challenge. Because to advancements in wireless communications, micro-electromechanical systems, and digital electronics, low-cost sensor devices, tiny may now be mass-produced. Storage space, Size, processing power, battery life, range, and data speed are all factors that define these gadgets. These devices are well suited for large-scale monitoring and control of physical events because of the range of transducers that may be attached to them. In [2], the authors of a clustering mutual coordination detection model that accounts for the quirks of wireless sensor networks reported their work. The percentage of accurate detections and the number of false positives have both gone up. Unknown assault detection is also quite poor. Intrusion detection is evolving in response to the rise of networked environments. While intrusion detection's initial focus was on single systems [3], the industry has recently switched its emphasis to networks [4]. In Network-oriented Intrusion Detection Systems, the source of events (and the object of analysis) is a distributed system consisting of many hosts and network links (NIDSs). Threats affecting several hosts and the network as a whole can be detected by NIDSs. The widespread adoption of WSNs in recent years has greatly aided human endeavors. While there are many advantages to WSN, there are also new dangers to people's safety and property. This has led to an increase in attention paid to the problem of "sensor network security" in the discipline of computer science. The purpose of network intrusion detection systems (IDS) is to, as the name implies, detect intrusions into a network [5][6]. Intrusion detection systems (IDS) are designed to protect sensitive data and physical assets by monitoring and analyzing data from a system's or networks most vital nodes in the event of an attack. Using a network-based intrusion detection system (NIDS) to monitor and analyze network traffic is essential for protecting a system from network-based attacks. Each incoming packet is inspected by a network intrusion detection system (NIDS) for irregularities. Depending on the severity of the threat, the system may provide an alert to administrators or prevent access to the network from the offending IP address. Snort is a network intrusion detection and prevention system developed as open-source software (NIPS). Snort is a multi-purpose network intrusion prevention system that may also act as a packet recorder

and sniffer. Using wireless networks, the sensor nodes collect data and relay it to a sink or base station. When compared to traditional networks, WSNs have severely constrained resources in the areas of energy, communication range, and computational capacity. Thus, in the design of WSNs, efficiency and data centralization are given high importance [7].

II. LITERATURE SURVEY

Several research has been done for Intrusion detection system in wireless sensor networks. [8][9] described a flood attack. In order to deplete bandwidth and node resources, an attacker may send a large number of Routing Request packets or other attack data throughout the network. His presentation of neighbor suppression [10] offered a solution to the MANETs flooding attack. We looked at how DoS attacks affect wireless networks [11]. In order to detect red and gray hole attacks in ad hoc networks, [12] presented cross-layer detection as an adaptive technique. In addition to the finite state machine, artificial immune system, and distributed intrusion detection [12], he also presented other methods of detecting intrusions. Host-based intrusion detection systems (HIDS) monitor and alter the actions of their host machine in order to spot suspicious behavior [13][14]. It's common for them to be curious about things like host traffic, application activity, configuration changes, and endpoint security. Monitoring network traffic for malicious activity that could compromise the network is the job of network-based intrusion detection systems (NIDS). An HIDS monitors both incoming and outgoing data from a device, and if it detects anything odd, it notifies the user. The evaluation of IDS in WSNs is discussed in further depth below, and concrete instances of issues are provided when necessary. Humidity, temperature, noise, pressure, pollution, and other environmental factors were all measured and evaluated with the use of WSNs. [15] analyzed the WSN's intrusion detection system. The use of anomaly-based, signature-based, and cross-layer IDSs were also investigated. This work made use of all three of the major components of an IDS: monitoring, analysis and detection, and alert. [16] proposed a novel rule-based intrusion detection system for WSNs. The major goal of this strategy is to prevent any security breaches that could compromise the WSN. In the knowledge-based intrusion detection described in, support vector machines were employed to label features as normal or abnormal [17]. The key benefits of this research model are its higher detection accuracy and lower classification error. In [18], the authors describe an intrusion detection system built on top of an artificial neural network, which makes use of a wide variety of features to tell the difference between typical and malicious network activity. Measurements of performance reveal that artificial neural networks outperform conventional methods when used to classify the retrieved features. The majority of techniques for implementing a reference monitor revolve around the security subsystem simultaneously scanning the code executed by each processor and comparing the results to a specified model. Code modification attacks can be detected by hardware-assisted security monitors [19][20] by monitoring run-time deviations in program execution and comparing them to a static model. The work of [19] demonstrates the usage of hash-based patterns for simpler blocks with fewer instructions. [20] presented a similar security method for multi-core computers, which also makes use of a monitoring software running on its own core. A good technique to prevent unauthorized parties from accessing or altering sensitive information is to encrypt it. Nevertheless, these techniques are not feasible for low-power embedded systems because of resource constraints and significant power consumption overheads. For WSN intrusion detection, researchers have traditionally used a method called single-point independent detection, in which the detection algorithm is created and executed on a single detection node (SID). An intrusion is established by analysis of intrusion detection once the proportion of failures attributed to rule violations is greater than the proportion of failures attributable to random network factors. Using linear prediction theory, [21] constructed a Markov mathematical prediction model for a single sensor node. If the absolute difference between actual and expected network traffic is greater than the configured threshold value, an attack behavior has been detected. Because nodes in WSNs have such low resources, it is not feasible to do single-point independent detection. The ELM algorithm's random starting point makes it challenging to construct a nonlinear model using data samples alone. As a solution, KELM is highly robust to changes in the model's parameters. Online kernel extreme learning machine modeling using rapid leave-one-out cross-validation was proposed by Zhang et al. in 2014. Although the experimental findings reveal that the random selection of dataset has a substantial impact on the classification performance, they also show that the suggested technique enhances the detection rate of the original kernel extreme learning machine. [6] looked at a wide variety of machine learning techniques that work well for IoT intrusion detection. Current work in IoT security research were examined, with an emphasis on Intrusion Detection Systems that make use of machine learning. Protocols, intelligent methods (such as machine learning), and accuracy attained in recent studies were the primary foci of this assessment. Finally, the challenges and promising futures of IoT security study were highlighted. [5] proposed an ANN-based Intrusion Detection System with less features. Key characteristics were extracted from the KDD Cup'99 dataset using the Mutual Information based feature selection technique. Comparing Mutation Information's performance against that of Mutual Information and ANN, it was shown to be superior, with a decreased false positive rate and no false negatives. Although this method is helpful, it requires more iterations of calculations to produce reliable findings. [22] demonstrates one common strategy for finding several matching string patterns. We presented theoretical and experimental data, as well as examined and discussed the algorithms. [23]devised an Efficient Exact Single Pattern Matching (EESP) technique to save pre-processing time and find all occurrences of the Pattern in a single, large text string. Bidirectional Pattern is an example of EESP. Using ideas from probability theory and Markovian queuing theory, [24] proposed a new type of ANN called RNN. Because its neurons can be

represented by simple counters, RNN can be easily implemented on hardware [25].

III. PROPOSED SYSTEM

Each of these datasets has a huge number of features, thus feature selection is performed to focus on the most relevant ones. Pre-processing entails procedures like normalizing and discretizing the data. To determine if the carefully chosen characteristics actually do improve detection accuracy, this proposes can test the system by looking into several classification techniques. Afterwards, it is put to use for model training.

A. Preprocessing of Data

This is step one of the processes. It is the goal of pre-processing to convert raw data into a more manageable and efficient form for further processing operations. The minmax technique is used for initial data normalization. When all training data, for instance, are on the same scale—say, between 0 and 1—normalization might reduce the amount of time needed for training. The suggested method achieves this by enforcing (1), where Z_{norm} is the normalized value and Z is the unnormalized starting point. Here, Z_{max} and Z_{min} stand for the highest and lowest possible values of each characteristic.

$$Z_{norm} = \frac{Z - Z_{norm}}{Z_{max} - Z_{min}} \tag{1}$$

Our second pre-processing technique is the Minimum Description Length (MDL)-based data discretization [26]. To discretize data means to transform a continuous data property into a set of discrete intervals while preserving as much information as possible about the original data.

B. Feature Extraction

In the Particle Swarm Optimization (PSO), particles stand in for individuals in an evolutionary algorithm and are used to perform searches (EA). In the beginning, a population of particles is produced at random. Each particle's position, denoted by the xi component of the position vector, stands in for a different possible solution. Several particles, each having its own velocity described by the vector w_z, buzz around the problem space. With y_z as input, a function $g_z$is computed that represents a quality metric at each time step. Each atom remembers its optimal location in a vector q_z, which is linked to its highest level of fitness. Moreover, the population's best particle position (denoted by q_h) is recorded [27]. There is also a localized variant of PSO that remembers where a particle is best situated relative to its topological neighbors.

Particle $z's$ velocity is recalculated at each time step s by utilizing both its local best position, q_z, and its global best position, $q_z(s)$.

$$w_z(s+1) = w_z(s) + d1\emptyset_1(q_z(s) - y_z(s) + d_2\emptyset_2(q_h(s) - y_z(s)) \tag{2}$$

Where $\varphi_1 \; and \; \varphi_2$ are independent random numbers in the interval [0,1], and the constants $d_1 \; and \; d_2$ are positive. The variable vi can only go as high as vmax. If the speed exceeds that threshold, it is corrected. The particle z can perform a search around its best possible positions q_z and q_h by modulating its velocity in this way. Each particle's position is updated according to the following equation using the new velocities:

$$y_z(s+1) = y_z(s) + w_z(s+1) \tag{3}$$

C. Feature Selection

Subsets of features are ranked by the Correlation Feature Selection (CFS) metric, which operates under the following premise: Two ideas stem from this premise. Correlations between features and classifications (q_{de1}) and between features themselves ($q_{e_ie_j}$) are two examples [28]. The classical linear correlation and the information-theoretic correlation are the two main metrics of the degree to which two random variables are linked. Correlation between features e_i and e_j is denoted as $q_{e_ie_j}$, while the correlation between features e_i and e_j is, as the name implies, the correlation between two features. The value of a q-feature subset p is given by the following equation, which is used in

$$Merit_p(l) = \frac{l\overline{q_{de}}}{\sqrt{(l + l(l-1)\,\overline{q_{ee}}}} \tag{4}$$

The average correlation between features is denoted by $\overline{q_{ee}}$, and the average correlation between classes by $\overline{q_{de}}$, as shown below:

$$\overline{q_{de}} = \frac{\sum_{i=1}^{l} q_{de1}}{l} \tag{5}$$

$$\overline{q_{ee}} = \frac{2\sum_{i,j=1,i\neq j}^{l} q_{e_ie_j}}{l(l-1)} \tag{6}$$

As a result, the equation may be rewritten as:

$$Merit_p(l) = \frac{\sum_{i=1}^{l} q_{de1}}{\sqrt{k + 2(\sum_{i,j=1,i\neq j}^{l} q_{e_ie_j})}} \tag{7}$$

Since the proposed have already normalized the data, equation (7) represents Pearson's correlation coefficient. It demonstrates that the correlation between the feature subset P and the target variable e depends on the number l of features in the subset P, the strength of the inter-correlations among those features, and the strength of the correlations between the features and e. The following inferences can be made using equation (7): The significance of the relationship between the feature subset p and the target variable e grows in proportion to the strength of the correlations between the features of the subset p and the features of the target variable e.

D. Training the Model:

1) GRU Model

Unlike standard neural networks, which are based on the weight connection between the layers, RNNs are a type of artificial neural network that is well-suited to the analysis and processing of time sequence data. The output

of an RNN takes into account both the current state and its memory of the past, as the former is stored in hidden layers. where $z(s)$ and $\hat{x}(s)$ stand for the input and output at time s, $c(s)$ stands of single layer of hidden for the output at time s, and $v_{cc}(s), v_{cz}(s), v_{cx}(s)$ stand for the weight matrices of the input, the hidden layer and the output, respectively.

$$c(s) = h_1(v_{cc}\, c(s-1) + v_{cz}\, z(s-1) + d_c \qquad (8)$$

$$\hat{x}(s) = h_2(v_{cx}\, c(s) + d_x \qquad (9)$$

Where d_c and d_x stand for the input and output bias vectors of a single hidden layer. Nonlinear activation function is denoted by h_1, and h_2 respectively [29]. Although RNNs excel when their outputs are close to their inputs, the gradient vanishing problem arises when there is a large number of weights and a prolonged time gap. The problems with the vanishing of gradients and RNN's straightforward hidden layer layout inspired the development of a new type of neural network called a GRU. GRU has fewer training parameters and only reset and update two gates. While LSTM has three (input, forget, and output). As a result, GRU achieves training convergence more quickly than LSTM. Where μ and tanh are activation functions, $e(s-1)$ is the input to the current unit and the output of the previous unit, and $e(s)$ is the output of the current unit and the input to the next unit. $z(s)$ are the training data inputs, and $\hat{x}(s)$ are the outputs of this unit as produced by the activation function. The candidate activation $\tilde{e}(s)$ is calculated in the same way as in a conventional recurrent unit, with p_w and p_q standing for the gates that can be reset and updated. The second gate is the update gate in GRU, which is responsible for bringing past data into the present. If the current state and the saved state should be combined, the reset gate is utilized to make that determination, and the value of u can vary from 0 to 1 (with 0 retaining the most information). The smaller the value of r, the more data from the past is disregarded, with a range of -1 to 1.

$$\begin{aligned} p_w &= \mu(v_w[e(s-1), z(s)] + d_w), \\ p_q &= \mu\big(v_q[e(s-1), z(s)] + d_q\big), \\ \tilde{e}(s) &= \tanh\left(v_e\big[p_q * e(s-1), z(s)\big] + d_e\right), \\ e(s) &= (1-p_w) * e(s-1) + p_w * \tilde{e}(s) \end{aligned} \qquad (10)$$

where v_w, v_q and v_e stand for the weight matrices learned during training on the update gate, reset gate, and candidate activation $\tilde{e}(s)$, while d_w, d_q and d_e are the corresponding bias vectors.

2) Convolutional Neural Network (CNN)

One variety of ANN that excels with high-dimensional data is the CNN. It finds extensive application in text categorization, video indexing, and image processing. Modern sensors and other apparatus are widely distributed throughout the electricity network. By basing its information on the location of the sensors and the time sequence, the spatiotemporal matrix was designed to preserve the locational features of data acquired with help of devices and sensors of intrusion system. In this case, this proposes present the spatial-temporal matrix as

$$z = \begin{matrix} z_1(1) & z_1(2) & z_1(m) \\ \vdots & \vdots & \vdots \\ z_l(1) & z_l(2) & z_l(m) \end{matrix} \qquad (11)$$

Where l stands for the lth intelligent sensor, m for the m^{th}time sequence, and $z_l(m)$ for the information gathered by kth sensor of instant n. The intrusion feature was extracted by using CNN to analyze the matrix of spatiotemporal. Figure 1 represents the CNN's architecture.

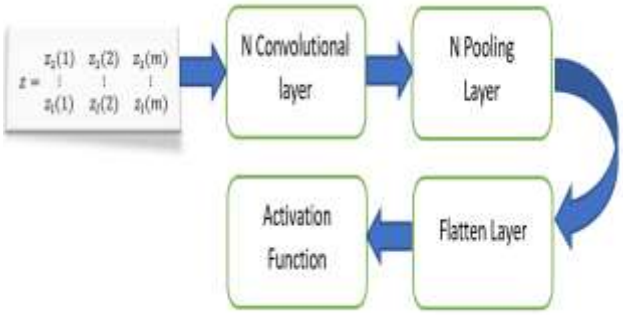

Fig. 1. Architecture of CNN

Figure 1 depicts the first step, which entails stacking several two-dimensional spatiotemporal matrices into three-dimensional matrix blocks before applying a convolution operation to the resulting matrix. The convolution process is used to obtain a highly abstract feature, and the pooling operation is then applied to the convolution operation's outputs. The pooling procedure allows neural network parameters to be lowered by decreasing the matrix size and the node numbers without affecting the input matrix's depth. The highly abstract feature was generated by iterative convolution and pooling techniques, and a one-dimensional vector is flattened for easier inclusion into the fully linked layer. Hence, the value of bias and weight of the fully connected layer can be found through a series of repeated calculations. The activation function's output is what provides the prediction results in the end.

3) CNN-GRU Model

The GRU-CNN network is a hybrid technique that takes advantage of the strengths of both the GRU module (which is great at sequence data processing time) and the CNN module (which is great at dealing with data which is at high dimensions). The reported GRU-CNN hybrid networks use spatiotemporal matrices and time sequence data reflecting data from intrusion detection to make predictions about future values. Two-dimensional input, such as images and spatiotemporal matrices, play to the CNN module's strengths. Using the pooling and convolutional layer, the CNN module effectively represents the data from the spatiotemporal matrices by taking use of the connection which is local and the weights of shared to directly extract local features. Each of the two convolution layers in the

CNN module is built with its own pooling, convolutional, and flatten operation. A second pooling operation is used to reduce the high-dimensional data to the dimension of interest, and the fully connected layer is then linked to the CNN module output. While forget gate would clear out any unnecessary information, the memory cell would allow GRU to glean insights from previous data over time. The GRU module receives input sequences in real time and has many gated recurrent units whose outputs are connected to the fully connected layer. By averaging the output of the neurons in the fully linked layers, this proposed can finally acquire the load prediction findings.

IV. RESULT AND DISCUSSION

Intrusion Detection Systems (IDSs) have developed into a crucial component of modern security infrastructure due to their role in risk management. Because of its similarities to other pattern recognition systems, feature extraction is a crucial part of any effective intrusion detection system (IDS). In this case, the features are extracted using CFS, the method proposed. In the end, it's put to use in model training.

To what extent it alters the network weights in response to the loss gradient is determined by the learning rate, a hyperparameter in the deep learning model. This article starts by choosing a range of values for the learning rate to compare, as shown in Figure 2, which shows the MC-GRU models detection accuracy at different learning rates. The experimental results are summarized in Figure 2, which shows that the model achieves a high detection accuracy of 0.9873 at a learning rate of 0.002. As a result, the article uses a learning rate of 0.002 for the model.

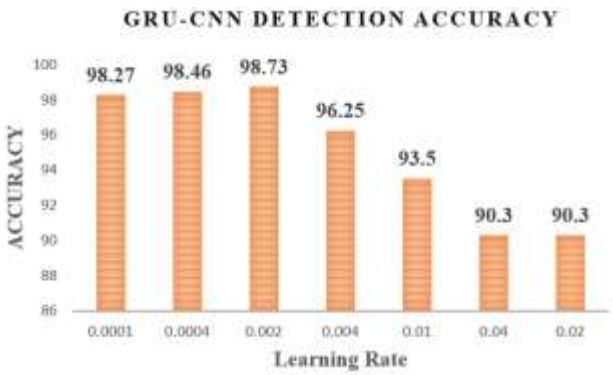

Fig. 2. Different Learning Rates for GRU-CNN

Compared to CNN (92.54%) and GRU (94.62%), the accuracy of our proposed GRU-CNN based intrusion detection in wireless sensor network technique is significantly improved. Our suggested CNN-GRU based intrusion detection in wireless sensor network approach has been shown to succeed where others have failed in a number of experiments.

TABLE I. COMPARISON OF THE MODELS

Models	ACCURACY	PRECISION	RECALL
CNN	92.54	90.83	86.62
GRU	94.62	92.83	89.96
CNN-GRU	98.73	97.29	95.67

Table I displays the chosen models (CNN, GRU, and GRU-CNN) because to their tolerable accuracy. The GRU-CNN algorithm model outperformed the other two in the aforementioned experiment.

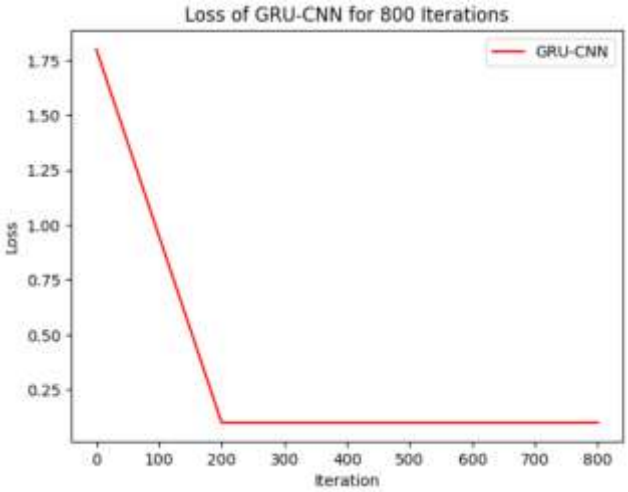

Fig. 3. Loss of GRU-CNN for 800 Iterations

Figure 3 shows that after 200 iterations, the GRU-CNN model has achieved rapid convergence, with loss values smaller than 0.2. After 800 iterations, the loss values for all these approaches are extremely low, indicating that they all learn the training set effectively and produce satisfactory test-set results. After training the training set for almost 800 iterations, it was discovered that the loss value fell very slowly. Although these models perform admirably on the test set, as the number of iterations is raised, their predictions on the training set deteriorate. Since it was clear that the models were overfit, the experiment's iteration times were set at 800 iterations.

V. CONCLUSION:

Due to their deployment in chaotic and potentially dangerous conditions, Wireless Sensor Networks (WSNs) are prone to malfunctions. Because of this, problems with software, hardware, and communication are common in WSN. Intrusion detection in WSNs is difficult because of the sensors' limited resources and the wide variety of application domains. There are several different kinds of assaults that are frequently launched against networks. More than two decades of uninterrupted success in preventing network attack can be attributed to the growing sophistication of AI algorithms. In order to train a model, the proposed method first applies normalization and discretization to the input data before moving on to feature extraction with PSO (Particle Swarm Optimization), feature selection with CFS, and ultimately model training with CNN-GRU. The proposed method in this investigation suggests a neural network that integrates characteristics of the CNN and the GRU model. The GRU component is responsible for sequence date time processing, while CNN component is responsible for spatiotemporal matrices processing. It outperforms the

other two models such as CNN and GRU which produces an accuracy of about 98.7%.

REFERENCES

[1] C. Chepken, E. Blake, and G. Marsden, "Day labour mobile electronic data capture and browsing system," *ACM Int. Conf. Proceeding Ser.*, vol. 91, no. 8, pp. 69–76, 2011, doi: 10.1145/2072221.2072230.

[2] X. Sun, B. Yan, X. Zhang, and C. Rong, "An integrated intrusion detection model of cluster-based wireless sensor network," *PLoS One*, vol. 10, no. 10, pp. 1–16, 2015, doi: 10.1371/journal.pone.0139513.

[3] D. E. Denning, "An Intrusion-Detection Model," *IEEE Trans. Softw. Eng.*, vol. 13, no. 2, pp. 222–232, 1987.

[4] B. Mukherrjee, T. Heberlein, and K. Levitt, "Network intrusion detection," *IEEE Network*, vol. 26–41, no. May/June. 1994.

[5] G. K. P and D. D, "Intrusion Detection Using Artificial Neural Network With Reduced Input Features," *ICTACT J. Soft Comput.*, vol. 1, no. 1, pp. 30–36, 2010, doi: 10.21917/ijsc.2010.0005.

[6] K. A. P. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. de Albuquerque, "Internet of Things: A survey on machine learning-based intrusion detection approaches," *Comput. Networks*, vol. 151, pp. 147–157, 2019, doi: 10.1016/j.comnet.2019.01.023.

[7] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: A survey," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 636–641, 2009, doi: 10.1109/WAINA.2009.192.

[8] P. Yi, T. Zhu, Q. Zhang, Y. Wu, and J. Li, "A denial of service attack in advanced metering infrastructure network," *2014 IEEE Int. Conf. Commun. ICC 2014*, pp. 1029–1034, 2014, doi: 10.1109/ICC.2014.6883456.

[9] P. Yi, Y. Hou, Y. Zhong, S. Zhang, and Z. Dai, "Flooding attack and defence in Ad hoc networks," *J. Syst. Eng. Electron.*, vol. 17, no. 2, pp. 410–416, 2006, doi: 10.1016/S1004-4132(06)60070-4.

[10] P. Yi, Z. Dai, S. Zhang, and Y. Zhong, "4," *Int. J. Inf. Technol.*, vol. 11, no. 2, pp. 83–94, 2005.

[11] P. Yi, F. Zou, Y. Zou, and Z. Wang, "Performance analysis of mobile ad hoc networks under flooding attacks," *J. Syst. Eng. Electron.*, vol. 22, no. 2, pp. 334–339, 2011, doi: 10.3969/j.issn.1004-4132.2011.02.022.

[12] P. Yi, T. Zhu, N. Liu, Y. Wu, and J. Li, "Cross-layer detection for black hole attack in wireless network," *J. Comput. Inf. Syst.*, vol. 8, no. 10, pp. 4101–4109, 2012.

[13] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," *Proc. Netw. Distrib. Syst. Secur. …*, vol. 1, pp. 253–285, 2003.

[14] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection and monitoring through VMM-based," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, pp. 1–28, 2007.

[15] N. A. Alrajeh, S. Khan, and B. Shams, "Intrusion detection systems in wireless sensor networks: A review," *Int. J. Distrib. Sens. Networks*, vol. 2013, 2013, doi: 10.1155/2013/167575.

[16] T. Eswari and V. Vanitha, "A novel rule based intrusion detection framework for Wireless Sensor Networks," *2013 Int. Conf. Inf. Commun. Embed. Syst. ICICES 2013*, pp. 1019–1022, 2013, doi: 10.1109/ICICES.2013.6508172.

[17] J. Jacob I. and E. Darney P., "Artificial Bee Colony Optimization Algorithm for Enhancing Routing in Wireless Networks," *J. Artif. Intell. Capsul. Networks*, vol. 3, no. 1, pp. 62–71, 2021, doi: 10.36548/jaicn.2021.1.006.

[18] F. Cauteruccio *et al.*, "Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance," *Inf. Fusion*, vol. 52, pp. 13–30, 2019, doi: 10.1016/j.inffus.2018.11.010.

[19] S. Mao and T. Wolf, "Hardware support for secure processing in embedded systems," *IEEE Trans. Comput.*, vol. 59, no. 6, pp. 847–854, 2010, doi: 10.1109/TC.2010.32.

[20] M. K. Yoon, S. Mohan, J. Choi, J. E. Kim, and L. Sha, "SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems," *Real-Time Technol. Appl. - Proc.*, pp. 21–32, 2013, doi: 10.1109/RTAS.2013.6531076.

[21] W. Zhang, D. Han, K. C. Li, and F. I. Massetto, "Wireless sensor network intrusion detection system based on MK-ELM," *Soft Comput.*, vol. 24, no. 16, pp. 12361–12374, 2020, doi: 10.1007/s00500-020-04678-1.

[22] K. Prabha and S. Sukumaran, "Improved Single Keyword Pattern Matching Algorithm for Intrusion Detection System," *Int. J. Comput. Appl.*, vol. 90, no. 9, pp. 26–30, 2014, doi: 10.5120/15604-3625.

[23] A. Islam Jony, "Analysis of Multiple String Pattern Matching Algorithms," *Int. J. Adv. Comput. Sci. Inf. Technol.*, vol. 3, no. 4, pp. 2296–1739, 2014.

[24] E. Gelenbe, "Random Neural Networks with Negative and Positive Signals and Product Form Solution," *Neural Comput.*, vol. 1, no. 4, pp. 502–510, 1989, doi: 10.1162/neco.1989.1.4.502.

[25] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1071–1083, 2002, doi: 10.1109/TCSVT.2002.806808.

[26] T. Ahmad and M. N. Aziz, "Data preprocessing and feature selection for machine learning intrusion detection systems," *ICIC Express Lett.*, vol. 13, no. 2, pp. 93–101, 2019, doi: 10.24507/icicel.13.02.93.

[27] Y. Chen, A. Abraham, and B. Yang, "Feature selection and classification using flexible neural tree," *Neurocomputing*, vol. 70, no. 1–3, pp. 305–313, 2006, doi: 10.1016/j.neucom.2006.01.022.

[28] H. T. Nguyen, K. Franke, and S. Petrovic, *Feature Extraction Methods for Intrusion Detection Systems*, no. May 2014. 2012. doi: 10.4018/978-1-4666-0978-5.ch002.

[29] L. Wu, C. Kong, X. Hao, and W. Chen, "A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model," *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/1428104.