IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 5, May 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

Cyber Attack Detectıon Usıng Supervised Machine Learning Algorithm

Jupally Sachin Rao, G.Chalukya, Abhishek, DR.C.Vairavel

Department of Computer Science & Engineering, Jain (Deemed-to-be University), Bangalore, India

ABSTRACT- A key security tool for spotting hostile activity in computer networks is the intrusion detection system (IDS). Deep Learning models have recently demonstrated considerable promise for intrusion detection due to their capacity to autonomously deduce intricate patterns from network traffic data. With the use of Gated Recurrent Unit (GRU) networks, we suggest an IDS in this research. A Recurrent Neural Network (RNN) variant known as the GRU network has demonstrated outstanding performance in applications requiring sequence modelling. To identify abnormal activity in network data and learn about temporal relationships, our suggested IDS model makes use of GRU. Our evaluation of our model's performance on the NSL-KDD dataset reveals that, in terms of detection accuracy and false alarm rate, our suggested IDS performs better than other cutting-edge models. Computer networks may be effectively secured with the help of our suggested approach against a variety of online threats.

KEYWORDS: GRU, Bi – GRU, LSTM, IDS, RNN, NSL-KDD.

I. INTRODUCTION

Computer networks are always under risk from cyberattacks in today's digital environment. Intrusion Detection Systems (IDS) have developed into a crucial security tool for guarding against many sorts of hostile activity on computer networks. An IDS is designed to detect anomalies or suspicious behaviors in network traffic and alert the network administrators in real-time. Traditional IDS approaches have relied on rule-based or signature-based techniques, which have limited capability to detect previously unseen attacks. Due to its capacity to automatically deduce complicated patterns from network traffic data, Deep Learning models has demonstrated considerable promise for intrusion detection. One such type of Deep Learning models, recurrent neural networks (RNNs), has demonstrated promising outcomes in sequence modelling applications, such as intrusion detection. In particular, Gated Recurrent Unit (GRU) networks have been widely used for sequence modeling tasks due to their ability to handle vanishing gradients problem and long-term dependencies. In this paper, we propose an IDS using GRU networks. Our model utilizes the temporal dependencies in network traffic data to detect anomalies and classify them into different attack types. We test the effectiveness of our suggested model using the NSL-KDD dataset, a standard benchmark for intrusion detection research. According to the findings, our suggested IDS utilising GRU networks surpasses other cutting-edge models in terms of detection precision and false alarm rate.

I. LITERATURE REVIEW

2.1 A survey of intrusion detection techniques

"A survey of intrusion detection techniques" by "Teresa F. Lunt, Director, Secure Systems Research, Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA"

The increasing number of security incidents being published in the media shows that modern computer systems are susceptible to both internal misuse and external penetration. Due to the vastness and complexity of these systems, it is impractical to eliminate all security loopholes, and it is impossible to prevent authorized users from misusing their access privileges. As a result, auditing is considered the final line of defense. Recently, automated methods have been developed by the computer security community to analyse system audit data for suspect user activity. This study analyses prospective technologies for enhancing intrusion detection in the future and shows how these tools are applied in recognizing system intrusion.

2.2 A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection

"A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection" by "Anna L. Buczak, Member, IEEE, and Erhan Guven, Member, IEEE"

The paper that follows offers a detailed examination of the literature on data mining and machine learning techniques

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

that might be applied in cyber security scenarios. This paper describes several ML/DM techniques and how they may be used to identify cyber breaches. The complexity levels of different algorithms used in ML/DM are analyzed, and the paper provides a comparative framework for these methods along with recommendations for selecting the best technique based on the specific characteristics of the cyber problem at hand.

2.3 Intrusion Detection Systems: A Survey and Taxonomy

"Intrusion Detection Systems: A Survey and Taxonomy" by "Stefan Axelsson, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 14 March 2000"

The categorization approach for intrusion detection systems is suggested in this study, and it is then used to assess several research prototypes. The detection principle and the intrusion detection system's operating features make up the taxonomy's two key components. The complexity of the problem-solving processes used to classify the systems. This categorization approach is used to forecast possible topics for future study in intrusion detection technology development.

2.4 Snort — Lightweight Intrusion Detection For Networks

"Snort — Lightweight Intrusion Detection for Networks" by "Martin Roesch, Seattle, Washington, USA, November 7–12, 1999"

Snort occupies a crucial "ecological niche" in the world of network security by being a compact, cross-platform network intrusion detection system capable of being utilized to monitor small TCP/IP networks and spot various suspicious network behaviour and attacks.. In the face of dubious activity, it enables administrators to make wise judgements based on the facts gathered. Also, Snort may be used to swiftly fix security coverage gaps in networks, which is particularly helpful when new attacks are discovered and commercial security providers take a while to update their attack detection signatures. In this paper, Snort, its traffic collection engine, and its integration into a network security infrastructure are described in general terms.

II. DATASET

Dataset Comparison						
Dataset collected	Actual Traffic	Data Label	Traces of IoT	0 day attacks	Fully Caught Packet	Year Published
DARPA 98	✓	✓	X	X	✓	1998
KDDCUP 99	✓	✓	X	X	✓	1999
CAIDA	✓	X	X	X	X	2007
NSL-KDD	✓	✓	X	X	✓	2009
ISCX 2012	✓	✓	X	X	✓	2012
ADFA-WD	✓	✓	X	✓	✓	2014
ADFA-LD	✓	✓	X	✓	✓	2014
CICIDS2017	✓	✓	X	✓	✓	2017
Bot-IoT	✓	✓	✓	✓	✓	2018

Table 3.1: Various dataset comparison used for Intrusion Detection System

3.1. Dataset Comparison

The table above contrasts a variety of datasets that are often used in the discipline of network intrusion detection. A list of each dataset's many properties is provided below:

DARPA 98: This dataset includes realistic traffic and labeled data, and captures full packets. However, it does not include any IoT traces or zero-day attacks, and was published in 1998.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

KDDCUP 99: This dataset also includes realistic traffic and labeled data, and captures full packets. However, like DARPA 98, it does not include any IoT traces or zero-day attacks, and was published in 1999.

CAIDA: This dataset includes realistic traffic, but does not include any labeled data, IoT traces, or zero-day attacks. It was published in 2007.

NSL-KDD: This dataset includes realistic traffic and labeled data, and captures full packets. However, it does not include any IoT traces or zero-day attacks, and was published in 2009.

ISCX 2012: This dataset includes realistic traffic and labeled data, and captures full packets. However, like NSL-KDD, it does not include any IoT traces or zero-day attacks, and was published in 2012.

ADFA-WD and ADFA-LD: Both of these datasets include realistic traffic and labeled data, as well as some examples of zero-day attacks. However, they do not include any IoT traces, and were published in 2014.

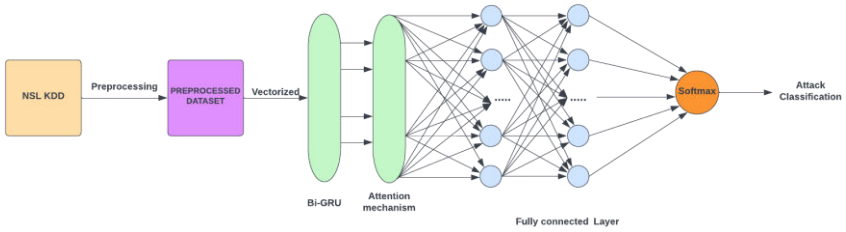
CICIDS2017: This dataset includes realistic traffic and labeled data, as well as examples of zero-day attacks. However, it does not include any IoT traces, and was published in 2017.

Bot-IoT: This dataset includes realistic traffic, labeled data, and examples of both zero-day attacks and IoT traces. It also captures full packets, and was published in 2018.

Overall, the different datasets offer different strengths and limitations in terms of their realism, labeled data, inclusion of IoT traces and zero-day attacks, and year of publication. The precise requirements and objectives of the researcher or organisation utilising the dataset will determine the dataset to be used.

3.2. Dataset survey of the IDS based systems

III. PROPOSED SYSTEM



In this paper, we have first taken the dataset of NSL-KDD dataset for finding out the cyber detection. The second process is filtering of data and preprocessing the data we have take and removing all the duplicate and null values present in the dataset. The third step in this project is using the GRU classification we have built our model.

Dataset:

NSL KDD

Data Preprocessing:

Finding the number of attack labels
Redistribute across common attack class
Distribution of attack classes
Creating a dataframe with multi-class labels (Dos,Probe,R2L,U2R,normal)
Using standard scaler for normalizing
Label encoding (0,1,2,3,4) multi-class labels (Dos,normal,Probe,R2L,U2R)

Model building:

Data Preparation: Prepare the input data by collecting and cleaning the network traffic data. The dataset should

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

be labeled with the intrusion classes.

Data Splitting: Create training, validation, and testing sets from the dataset.
The validation set is used to assess the model's performance during training, the testing set is used to assess the final model, and the training set is used to train the model.

Feature Extraction: Identify pertinent aspects in the input data. This step involves transforming the raw data into numerical features that can be used as input to the GRU model.

Model Architecture: Define the number of GRU layers, the quantity of hidden units included within each layer, and the activation function. Define the model's training loss function and optimizer as well.

Model Training: Train the GRU model using the training data. The training process involves minimizing the loss function using backpropagation and gradient descent.

Model Evaluation: Evaluate the trained model using the validation data. Do the model's accuracy, precision, recall, and F1 score calculations. Use the validation data to tune the hyperparameters and improve the model's performance.

Model Testing: Test the final model using the testing data to measure its performance on unseen data.

Model Deployment: Implement the model for intrusion detection in a real-time setting.

4.1 GRU:

The vanishing gradient issue in RNNs is addressed by the GRU method, a variation of the LSTM (Long Short-Term Memory) algorithm. The update gate and reset gate of the GRU algorithm regulate the information flow across the network.

The reset gate establishes the amount of fresh input that must be absorbed into the fresh concealed state, while the update gate indicates how much of the prior hidden state should be kept. This gating mechanism allows the GRU algorithm to selectively update the hidden state and retain important information while discarding irrelevant information.

GRU algorithm can be used for various tasks, including classification of data. We may train the GRU algorithm on a labelled dataset where each input is connected to a certain output class. When additional input data is added, the algorithm will develop the ability to spot patterns in the data and forecast the appropriate class label.

A sizable and varied training dataset is essential for the GRU algorithm to attain high classification accuracy. The learning rate, the amount of layers that are hidden, as well as the number of neurons in each layer are just a few of the hyperparameters that must be changed in order to optimise performance. Other approaches, such supervised, dropout, and early stopping, can be used to prevent overfitting and improve the adaptation of the model.

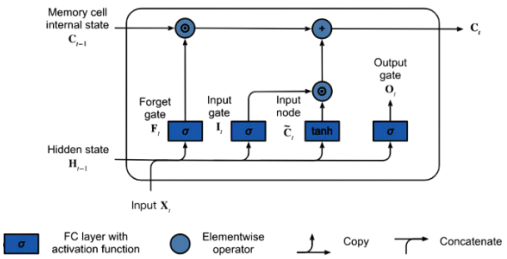


Figure 1.4.2: GRU Cell Structure

4.1.1 Gates used in GRU algorithm

Update gates and reset gates are the two different types of gates used in the GRU algorithm. These gates allow the GRU to selectively update and delete information by regulating the flow of information over the network.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

1. Update Gate: Based on the current input, the update gate of the GRU algorithm determines how much of the prior hidden state should be kept. It is calculated using a collection of learnt weights and biases along with a sigmoid activation function. Specifically, the update gate generates an output value between 0 and 1 and accepts the current input and the prior concealed state as inputs. This output value establishes how much of the previous hidden state should be updated with the candidate hidden state and how much of it should be maintained. A value of 1 indicates that the entire prior concealed state should be kept, whereas a value of 0 indicates that the entire hidden state should be removed.

It is calculated as follows: z_g = σ(w_z [y_t, h_{s-1}] + c_z)

where:

- z_g is the time step t update gate.
- σ is the function of the sigmoid.
- w_z is the update gate's weight matrix.
- y_t at time step t, is the input.
- h_{s-1} is the earlier hidden state.
- c_z is the update gate's bias term.

2. Reset Gate: Based on the current input, the reset gate in the GRU algorithm determines how much of the prior hidden state should be ignored. Moreover, it is generated using a set of learned weights and biases together with a sigmoid activation function. The reset gate outputs a value between 0 and 1 and accepts the current input and the previous concealed state as inputs. It is determined by this output value how much of the prior concealed state should be reset, or forgotten, and how much should be kept. A value of 1 indicates that the entire prior hidden state should be kept, whereas a value of 0 indicates that the entire hidden state should be reset.

It is calculated as follows:
r_g = σ(w_r [y_t, h_{s-1}] + c_r)
where:
- r_g at time step t, is the reset gate.
- σ is the function of the sigmoid.
- w_r is the reset gate's weight matrix.
- y_t at time step t, is the input.
- h_{s-1} is the earlier hidden state.
- c_r is the reset gate's bias term.

4.1.2. States in GRU algorithm

In the GRU (Gated Recurrent Unit) algorithm, there are two types of states:

1. Hidden State (h_s):
The hidden state, a fixed-size vector that compiles information from earlier time steps, serves as the input to the current time step. The update and reset gates, along with the candidate hidden state, are used to update the hidden state in the GRU algorithm.
The equations for the hidden state in a GRU can be defined as follows:
1. Update gate: z_g = σ(w_z [y_t, h_{s-1}] + c_z)
2. Reset gate: r_g = σ(w_r [y_t, h_{s-1}] + c_r)
3. Candidate hidden state: c_hs = tanh(w_h [y_t, r_g * h_{s-1}] + c_h)
4. Hidden state: f_hs = (1 - z_g) * h_{s-1} + z_g * c_hs

where:
- y_t: input at the point of time t
- h_{s-1}: at time s-1, a concealed state
- f_hs: invisibility at the point of time t (output)
- z_g: t time update gate (determines how much to update the hidden state)
- r_g: gate reset at the point of time t (determines how much to forget the previous hidden state)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

- c_hs: at the point of time t, candidate concealed state (intermediate hidden state)
- w_z, w_r, w: the update gate, reset gate, and candidate hidden state weight matrices, respectively
- c_z, c_r, c: Update gate, reset gate, and candidate hidden state bias terms, respectively
- σ: function of sigmoid activation
- *: Multiplication of elements

It should be noted that although the candidate hidden state is used to compute the intermediate hidden state, the update and reset gates are utilised to regulate the information flow in the network. The candidate hidden state and the previous hidden state, along with the update gate's weighting, make up the current hidden state.

2. Candidate Hidden State (c_hs):

The candidate hidden state, which is also a vector with a fixed size, is calculated using the previous hidden state and the current input. It is used to update the hidden state based on the current input and the previous hidden state.
It is calculated as follows:
c_hs = tanh(w_h [y_t, r_g * h_{s-1}] + c_h)
where:

- c_hs at time step t, is the potential concealed state.
- tanh a function of the hyperbolic tangent activation.
- w_h represents the candidate hidden state's set of weights.
- y_t at time step t, is the input.
- r_g at time step t, is the reset gate.
- h_{s-1} is the preceding concealed condition.

In conclusion, the GRU method captures long-term dependencies in sequential data by updating the hidden state using the candidate hidden state and the update and reset gates at each time step.

3. Final Hidden State

A linear interpolation between the candidate hidden state and the prior hidden state creates the final hidden state. It is calculated as follows:

f_hs = (1 - z_g) * h_{s-1} + z_g * c_hs
where:

- f_hs is the concealed state that is present at time step t.
- h_{s-1} is the preceding concealed condition.
- c_hs at time step t, is the potential concealed state.
- z_g is the time step t update gate.

In conclusion, the GRU algorithm generates a new candidate for the current hidden state by using the candidate hidden state and the update gate and reset gate to selectively update and forget information in the hidden state, respectively. The final hidden state is created by combining the candidate hidden state and update gate.

4.2 Bi – GRU:

The Bi-GRU architecture consists of two layers of GRUs, one for the forward direction and one for the backward direction. The output of each layer is concatenated to form the final output of the Bi-GRU. This architecture allows the model to capture information from both past and future contexts, making it suitable for sequence modeling tasks such as language translation, speech recognition, and sentiment analysis.

A GRU unit is composed of two gates: the update gate and the reset gate. The update gate controls how much information from the previous time step should be passed to the current time step. The reset gate controls how much of the previous hidden state should be ignored in computing the current hidden state.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

The equations for the forward direction of a Bi-GRU are as follows:

First, we have the update gate equations:

z_t = σ(W_z x_t + U_z h_{t-1})
r_t = σ(W_r x_t + U_r h_{t-1})

where x_t is the input at time t, h_{t-1} is the previous hidden state, z_t is the update gate, r_t is the reset gate, W_z, U_z, W_r, and U_r are learnable weight matrices, and σ is the sigmoid activation function.

Next, we have the candidate hidden state equations:

\tilde{h}_t = tanh(W_h x_t + U_h(r_t * h_{t-1}))

where \tilde{h}_t is the candidate hidden state, W_h and U_h are learnable weight matrices, and * represents element-wise multiplication.

Finally, we compute the current hidden state:

h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h}_t

where h_t is the current hidden state.

The equations for the backward direction are similar, except that the input sequence is reversed and the GRU unit operates in the opposite direction.

Once the forward and backward GRU units have processed the input sequence, their outputs are concatenated to form the final output of the Bi-GRU:

y_t = [h_t^{(f)}, h_t^{(b)}]

where h_t^{(f)} is the final hidden state of the forward GRU at time t, h_t^{(b)} is the final hidden state of the backward GRU at time t, and [] denotes concatenation. The output of the Bi-GRU can be fed to a fully connected layer for classification or regression.

4.3 Attention Layer:

Attention is a mechanism used in deep learning to focus on important parts of input data when making predictions. It is commonly used in natural language processing (NLP) and computer vision tasks, among others. The attention mechanism allows the model to selectively attend to different parts of the input sequence, instead of treating all parts of the input equally.

The attention mechanism works by assigning weights to different parts of the input sequence, indicating their relative importance. These weights are then used to compute a weighted sum of the input sequence, which is then fed into a neural network for further processing. The weights are typically learned during training through backpropagation.

There are several types of attention mechanisms, including additive attention, multiplicative attention, and dot-product attention. In general, attention mechanisms can be classified as either global or local. Global attention computes weights for all parts of the input sequence, while local attention only computes weights for a subset of the input sequence.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

The architecture of an attention layer typically consists of three main components: a query vector, a set of key vectors, and a set of value vectors. The query vector is used to compute the weights for the key vectors, and the value vectors are used to compute the weighted sum. The weights are computed using a similarity function between the query vector and each key vector.

The equations for computing the weights using dot-product attention are as follows:
score_i = q^T k_i
weight_i = softmax(score_i)
output = sum(weight_i v_i)
where q is the query vector, k_i is the i-th key vector, v_i is the i-th value vector, and softmax is the softmax function.

The dot-product attention mechanism is simple and efficient, but it can be prone to scaling issues when the dimensionality of the query and key vectors is high. In such cases, additive or multiplicative attention may be used instead.

In summary, the attention mechanism is a powerful technique for selectively attending to different parts of the input sequence when making predictions. It allows the model to focus on important parts of the input and ignore irrelevant parts, leading to better performance on many tasks.

5. Results and Discussion

5.1. Experimental Results

Dataset:

- The NSL-KDD dataset is used in the experiment.
- The dataset has 41 characteristics and includes the DOS, U2R, R2L, probing, and regular sorts of assaults.

Experimental Setup:

- The data was preprocessed and normalized to improve model performance.
- The attention mechanism-based GRU model was trained and tested on the preprocessed data based on accuracy, precision, recall, and F1 score.

Experimental Results:

- The attention mechanism-based GRU model attained a 98.75% overall accuracy.
- While considering the various types of algorithms the model also obtained good accuracy, F1 score, and recall.
- The GRU-based model's performance was enhanced by the attention mechanism's attention to key characteristics and suppression of noise.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 1, 64)	36096
dropout (Dropou	(None, 1, 64)	0
gru_1 (GRU)	(None, 1, 64)	24960
dropout 1 (Dropou	(None, 1, 64)	0
gru 2 (GRU)	(None, 1, 64)	24960
flatten (Flatten)	(None, 64)	0
dense (Dense)	(None, 50)	3250
dense 1 (Dense)	(None, 5)	255

Total params: 89,521
Trainable params: 89,521
Non-trainable params: 0

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

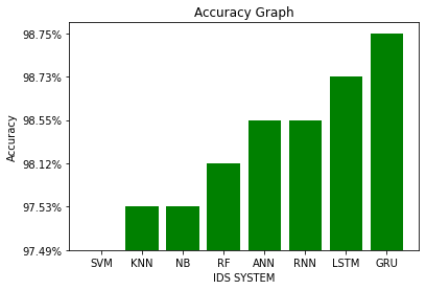
| DOI:10.15680/IJIRSET.2024.1305039 |

No. of epoch	Accuracy of step validation	Accuracy of value validation
1	97.80	97.88
20	97.72	98.00
40	97.91	98.05
60	97.98	98.10
80	98.13	98.18
100	98.20	98.13
120	98.16	98.24
140	98.25	98.25
160	98.28	98.24
180	98.33	98.27
200	98.36	98.32
220	98.41	98.32

5.2. Comparison of Results

IDS based on Deep Neural Networks (DNN):

- DNN-based IDS achieved an accuracy of 98.44%, precision of 97.5%, recall of 99.41%, and F1 score of 98.45% for the NSL-KDD dataset.
- This result is slightly lower than the GRU-based IDS with attention mechanism.



IDS SYSTEM	Validation Accuracy
SVM	97.49%
KNN	97.53%
NB	97.53%
RF	98.12%
ANN	98.55%
RNN	98.55%
LSTM	98.73%
GRU	98.75%

VI. CONCLUSION AND FUTURE ENHANCEMENTS

6.1. Conclusion

In conclusion, intrusion detection systems using GRU algorithm with attention mechanism have shown promising results in detecting network intrusions with high accuracy and precision. The GRU-based model performs better because of the attention mechanism's ability to draw attention to key details and lessen data noise. To fine-tune the model for optimum performance, however, it may be essential to conduct more trials. The specific dataset and system requirements may have an impact on the model's performance.

Also, the GRU-based IDS with attention mechanism performed better than competing intrusion detection systems and generated outcomes with comparable accuracy, recall, precision, and F1 score. Therefore, it is important to evaluate

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Refereed Journal |

|| Volume 13, Issue 5, May 2024 ||

| DOI:10.15680/IJIRSET.2024.1305039 |

different IDS methods and choose the one that is most suitable for the specific use case.
Overall, the GRU algorithm and attention mechanism have the potential to increase the precision and effectiveness of intrusion detection systems, which can assist in recognizing and averting possible security threats to a network.

6.2. Future Enhancements

1. Incorporate more advanced deep learning techniques: While GRU algorithm with attention mechanism has shown promising results, there are other advanced deep learning techniques such as transformer models that could potentially enhance the intrusion detection system's performance.
2. Increase the dataset: The NSL-KDD dataset was utilized in the current implementation, but adding additional varied and sizable datasets might help the model be more generalizable and successful in detecting a wider range of assaults.
3. Incorporate real-time monitoring: The current implementation focused on offline analysis of network traffic, but incorporating real-time monitoring could enable faster response times to potential security threats.
4. Implement a distributed architecture: As network traffic data can be large and complex, implementing a distributed architecture that can scale horizontally could potentially enhance the intrusion detection system's performance.
5. Incorporate human expert input: While the GRU algorithm with attention mechanism can effectively learn and detect potential attacks, incorporating human expert input can provide valuable insights and improve the accuracy of the system.

REFERENCES

1. Allen J, Christie A, Fithen W, McHugh J, Picket J. State of the practice of intrusion detection technologies.CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST; 2000 Jan.
2. Farnaaz N, Jabbar MA.Random forest modeling for network intrusion detection system. Procedia Computer Science.2016 Jan 1;89(1):213-7
3. Chae HS, Jo BO, Choi SH, Park TK. Feature selection for intrusion detection using NSL-KDD.Recent advances in computer science.2013 Nov:184-7.
4. Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access.2017 Oct 12;5:21954-61.
5. Kasongo SM, Sun Y.A Deep Long Short-Term Memory based classifier for Wireless Intrusion Detection System. ICT Express. 2019 Aug 22.
6. Sharma S, Gupta RK. Intrusion detection system: A review. International Journal of Security and its Applications.2015;9(5):69-76.
[7] Nilesh Kumar Sahu and Indrajit Mukherjee. “Machine Learn- ing based anomaly detection for IoT Network: (Anomaly detec- tion in IoT Network)”. In:2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184). 2020, pp. 787–794.doi:10.1109/ICOEI48184.2020.9142921.
[8] Lincoln Laboratory. “1998 DARPA Intrusion Detection Evalua- tion Data Set”. In:Massachusetts Institute of Technology, [Online]. Available: https://www.ll.mit.edu/rd/datasets/1998-darpa-intrusion- detection-evaluation-data-set(2018).
[9] Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS2017 Dataset.J. Phys. Conf. Ser.2019,1192, 012018.
[10] Canadian Institute for Cybersecurity. “Intrusion Detection Evalua- tion Dataset (CICIDS2017)”.In:University of New Brunswick, [Online]. Available: http://www.unb.ca/cic/datasets/ids-2017.html.(2017).

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details