International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(15s) 2026 pp. 120-131
https://doi.org/10.70917/ijcisim-2026-4352
Article

An Intelligent Hybrid Deep Learning Model for Proactive Cyber Threat Detection and Defense

Prasad A. Lahare[1], Manoj A. Wakchaure[2]

[1]MET Bhujbal Knowledge City, Institute of Engineering (MET BKC IOE), Nashik, Savitribai Phule Pune University (SPPU), Pune, Maharashtra, India.
Email: prasadlahare7@gmail.com
[2]Amrutvahini College of Engineering (AVCOE), Sangamner, Savitribai Phule Pune University (SPPU), Pune, Maharashtra, India.
Email: manoj13apr@gmail.com

Abstract: The research presents a hybrid AI based deep learning model that adds Bidirectional GRU (BiGRU) and Global Average Pooling (GAP) to detect software vulnerabilities. The main goal of this model is to find security issues in software source code, like SQLi and XSS attacks. The BiGRU helps in understanding the code by analysing it in both past and future directions, which improves the detection of patterns and relationships in the code. The GAP layer helps to reduce the size of the dense vector and prevents the model from over fitting. Overall, this approach improves the accuracy and efficiency of cyber threat detection. The researchers assess the proposed model using real-world vulnerability datasets which they collected from Kaggle and which contain labelled code samples of both vulnerable and non-vulnerable instances. The experimental results show that the proposed model performs well from all baseline models which include RNN, LSTM, GRU, and BiGRU because it achieves an accuracy rate of 97.5% and a precision rate of 98% and a recall rate of 97% and an F1-score of 97.5%. The framework provides perfect and scalable solution for early vulnerability detection which enables integration into DevSecOps pipelines while it improves software security through proactive identification of potential threats.

Keywords: Cyber threat detection, BiGRU, Global Average Pooling, Deep learning, Intrusion detection, Network security, Attack classification, Proactive defense

1. INTRODUCTION

Among no. of vulnerabilities, SQLi and XSS are the two primary and hazardous attacks according to OWASP. In these vulnerabilities, the attacker hacks the database query or places a malicious script in a web application that could possibly lead to data erosion or unauthorized access. Code-level diagnosis is critical to enhance software security and diminish risks in contemporary software production.

The global arena recognizes cyber security as the most critical issue. Technological threats have grown extensively impacting numerous organizations dependent on Internet-based operations. During the previous century onwards, the threat has been exponentially proliferating in its frequency. The question of uniqueness and the manner in which these operations continue to deteriorate as time passes are, of course, soon raised. Many studies suggest that countable amounts of dollars will be ransacked from cyberspace, which will have an enormous bearing on industries, government, and individuals across all strata[2].The typical approach to preventing intrusion nowadays would barely ever improve to enforce against zero-day vulnerabilities, as it would miss much more than if a novel attack were attempted. This further provides answers to the immediate demand for intelligent and adaptive security solutions. The trend in cyber threats has dramatically changed in the past decade. Hackers have now been using advanced persistent and polymorphic threats, or are further convincing others through well-designed social engineering, to bypass the traditional security checks [8]. Traditional rule based systems and statistical models are not flexible and capable to handle new and changing threats. Machine learning is used to solve the problem, but many traditional algorithms cannot understand the sequence of modern attack patterns and codes.

Deep learning methods, especially RNNs, work in proper manner when data that comes in sequences. Among these, GRU and BiGRU are good at learning patterns that depend on long-term relationships in the data. This study tries to improve existing and old intrusion detection systems, which still have some problems [4]. It proposes a hybrid model that combines BiGRU and Global Average Pooling (GAP). Firstly, the BiGRU looks at network data in both past and future directions, so it can detect the different types of attack patterns. Secondly, GAP layer is used to reduce the data size, storing only important information and preventing overfitting. This makes the model simpler and more proper. The system works step by step. First, it detects and classifies different cyber threats. Then, it automatically takes necessary action to handle them. This reduces response time and enhances security. Unlike traditional systems that depend on human decisions, this system works automatically and responds faster [1]. The main goal of this is to build a hybrid BiGRU-GAP-DENSE layer model for cyber threat detection, classification as well as prevention. This model is tested on many cyber-attacks it performs properly than the earlier model. It also improves the performance of the system because at the time of development all threats are wash out so it preserves its proactive defense nature.

2. LITERATURE REVIEW

In this present scenario n no. of Intrusion detection systems improve their nature. Earlier systems were based on traditional signatures, then moved to anomaly detection, machine learning methods. A study by Buczak and Guven highlights the drawbacks and limitations of traditional methods, such as high false alarms, difficulty in detecting new attack patterns and slow performance when handling huge datasets [2]. In the early days of intrusion detection, ML models like decision trees, SVMs, and random forests were used to detect and classify the cyber threat attack patterns. But one main problem with these models is that they struggle to specifically handle feature engineering and fail to capture how network traffic changes over the time.[8]. Manual feature extraction suffers much from a subjective bias, and most of the time, many such subtle patterns-the attacks that evolved solely sophisticated-went unnoticed. Saving the ML solution for the above-mentioned limitations. The CNNs were one of the first deep learning models to be applied in a network intrusion detection scenario, treating traffic data in the form of an image, in which an intrusion image was seen as distinct from a normal one [7]. Even if these developments on Convolutional Neural networks have shown considerable progress, it fundamentally works on spatial data and does not inherently possess the power to grasp the sequential nature of network events. This led researchers to take up the challenge to explore recurrent architectures that are specific for a more temporal data processing stream.

RNNs and their types, in particular the LSTM networks, are very appreciated for their capacity to learn long-term interdependencies in sequential data. Recent studies have proven that an LSTM-based intrusion detection system could possibly learn to identify the sophisticated attack pattern by establishing temporal relationships within a safe coding environment [9]. However, standard LSTMs remain computationally complex and difficult to train effectively for meaningful intrusion detection on large-scale data sets in cyber security applications. Though slightly less complex, Gated Recurrent Unit appeared to give the same performance as the LSTM in capturing the long-term decencies; they, however, had a greater number of parameters. Kim et al [4] mentioned that their GRU models yielded good results while consuming less time and computational resources for training than LSTM networks. Thus, in contrast to the LSTM architecture, only the GRU obtains good detection efficiencies in real-time threat detection situations by virtue of their simpler gating mechanism. The bidirectional recurrent architectures extend the capabilities of regular RNNs to process the sequences in both forward and backward directions. That helps considerably in enhancing the root-mean-square differences, as Sheikhan [6] showed in the case of intrusion detection by providing much greater temporal context. The bidirectional approach for network intrusion detection permits the ability of identifying attack patterns, which may not be very noticeable if one looks at the traffic flows through only one temporal perspective. Global Average Pooling G (GAP) is one of the few alternatives that are around today which tackle dimension penalization in the classification without loss of information that is actually good to preserve useful features. Unlike the Max pooling or Flatten operation methods, where GAP does all types of feasible structural regularization that would reduce over fitting [5]. In addition, the GAP method makes the structure of the network more interpretable, as now each feature map is associated specifically with an output image class.

There have been recent studies experimenting with hybrid architectures for enhancing the intrusion detection system. Apruzzese [1] introduced an ensemble approach combining several neural network architectures to increase robustness and generalization. These, however, are rare examples of hybrid models combining CNN and RNN through attention layers, and application-specifically-less of a BiGRU-GAP combination approach has already been reported in the literature, especially one that is dedicated to cyber-threat detection. Real-time threat detection and automatic response are still not fully studied in current research. Most of the studies mainly focus on improving accuracy and do not consider practical issues like performance, speed, scalability, and DevSecOps integration with real security systems [3].

Agrawal et al. used machine learning model to create a system that catches and blocks phishing scams and handle appropriately [10].Goodfellow et al. wrote a textbook that covers all the basic concepts of deep learning based model and by using this they aligned it with people and map how people use it in the real world [11].Kingma and Ba used Adam optimizer, which is a faster and smarter way for AI models to learn from raw data [12].Bhuyan et al. looked deeply at the different tools, techniques and methods tech experts use to spot behavior on different computer networks [13].Moustafa and Slay built the UNSW-NB15 dataset to give security researchers a fresh, modern way to test and validate out their anti-hacking tools [14].The Canadian Institute for Cybersecurity shared the CICIDS2017 dataset to give many researchers an original mix of normal internet traffic and actual cyberattacks to practice on [15].He et al. added intelligent and clever shortcuts to artificial neural networks, making it possible to train proper super deep AI models without breaking down them [16].LeCun et al. teamed up to write a big overview on how deep learning works and why it is changing the tech world [17]. Javaid et al. used a new deep learning terminology specifically made for catching hackers trying to break into networks [18].Wang et al. tested that AI can spot hidden malware just by looking at network traffic data like it is a regular image [19].Hochreiter and Schmidhuber invented the LSTM network to help computer systems to remember valid and important information in long chains of data without forgetting it [20].Cho et al. introduced a very new way for AI to translate text and understand human oriented languages accurately [21].Kipf and Welling created Graph Convolutional Networks to help AI to read data that is shaped like a connected network [22].Abadi et al. introduced TensorFlow, a software library built to run heavy AI workloads on lots of computers at single point of time [23].Chollet wrote a simple, hands on guide that teaches researchers how to code their own deep learning models using Python language [24].Krizhevsky et al. proposed a famous model called AlexNet that proved DNN are amazing at recognizing images [25].Hindy et al. list out modern digital threats into a very clear categories and reviewed the best tools available to finish them [26].

3. METHODOLOGY

3.1 Proposed Framework Architecture

Our Hybrid BiGRU-GAP-Dense model takes application security data, finds sequence patterns from this data, and then detects and classifies cyber-attack patterns known as cyber threats. After that, it can also help in taking the right and appropriate action against those cyber threats. Firstly, the raw data from publicly available datasets is preprocessed so it can be used by the DL model. In this step, important information is selected and arranged into structured fixed size sequence tokens. This includes taking features from tokens, joining related data flows, and splitting the data into fixed time based sequences. These sequences show how the activity changes over time and help in better threat detection.

The core of our architecture involves the use of a Bidirectional GRU layer to analyse these code tokens simultaneously in past and future directions. As the sequences are processed, each GRU cell preserves a hidden state that captures temporal dependencies, which are resistively governed by the update gate (determining how much new information must be input back into the cell) & the reset gate (determining how much information from the past should be forgotten), while allowing the model to look at both the forward and the backward contexts in each & every time step, providing extensive temporal awareness that is vital for the identification of attack patterns that may span longer time windows.

Following BiGRU layer comes GAP which is used to minimize the dimensionality of new representations. Instead of flattening output from the BiGRU or using max pooling, GAP computes the spatial average for each feature map thereby generating feature vectors with a fixed length independent of the length of the input sequences. This operation has a no. of benefits such as reducing the complexity of the model, providing an implicit layer of relaxation against overfitting, and making everything interpretable by ensuring that the learned features retain their semantic meanings.

Pooling the features will pass to a series of fully connected layers that will output final classification as per the input. For our dense layers, we implement dropout. Through validation, we decide on the dropout rate for dropout regularization, further reducing overfitting. The output layer that uses SoftMax separately classifies each of the threat categories: benign or normal traffic and the different kinds of attacks at the receiving end.

3.2 Data Prefiling

Source code samples, after being purged from duplicates, are processed for meaningful representations. The pre-processing pipeline comprises source code tokenization and normalization, besides turning them into numerical sequences with an embedding of source-code semantics. Each code sample is a sequence of tokens made to represent

syntactic and semantic elements of the code. These sequences are then padded to a fixed-length and fed into the BiGRU model for sequential learning. Normalizing features aims to standardize the features by getting them to mean zero and variance one, thereby stopping the full range of a feature from dominating the learning process. Normalization parameters are computed from training data only, and then the same transformations are applied to the validation and test sets to prevent data leakage. This is significant for gradient-based optimization in deep neural networks.

3.3 Training Process and Optimization

Model training makes use of supervised learning along with labelled network traffic dataset in which each observation is tagged with its corresponding ground truth label. The label informs whether the corresponding sequence beneath is to be classified into benign or specific attack categories. The loss function of use is cross entropy with the categorical implementation natural for the problem of multi class classification. The Adam optimizer is used to select the work out appropriate parameters in order to update adaptively across configurations, providing each parameter with an adaptive learning rate and enable the learner to efficiently account for situations where sparsity in gradients is a common occurrence, as often seen in a work environment configured in a sequential manner. For stopping the overfitting and cutting down on continuous training, early stopping mechanism was instituted on validation loss metric to avoid the infeasibility of upskilling. At passing through each epoch, validation performance will be checked and if at all, for a certain count of consecutive epochs, the validation loss is seen to have failed to reduce, the training ends. This protocol will apply the solution to illustrate the extent of generalization of the model in contrast to merely being a regurgitation engine wherein it just memorizes examples.

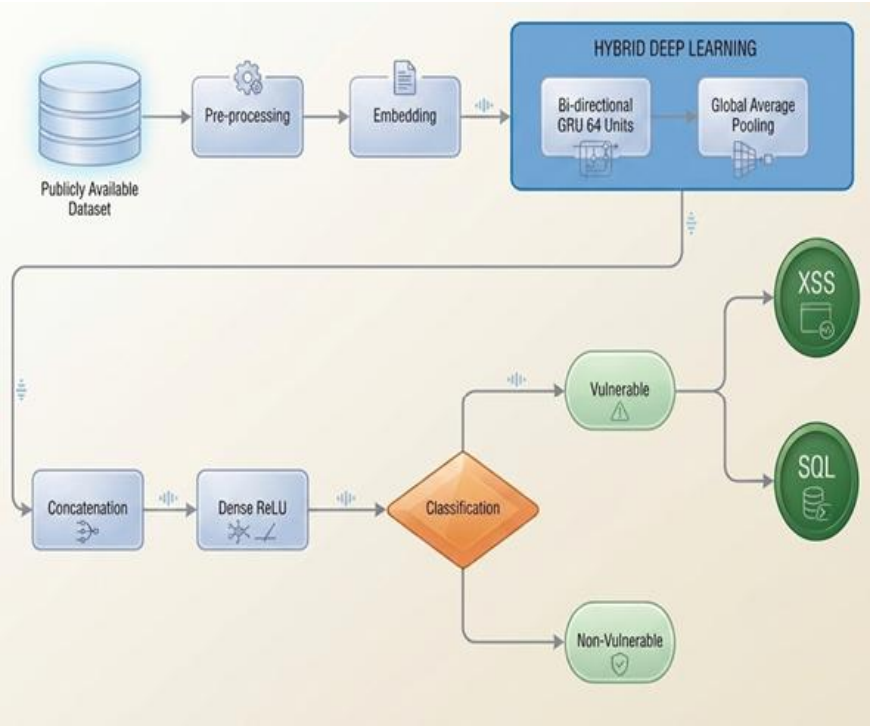

Figure 1: Hybrid Bigru–Gap–Dense Architecture

3.4 Algorithmic Representation of Proposed BiGRU-GAP Framework

Input:

Network traffic dataset $D = \{x_1, x_2, ..., x_n\}$

Labels $L = \{Benign, DoS, DDoS, PortScan, Brute Force, Web Attack\}$

Output:

Predicted labels $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$

1: Initialize feature extraction module

2: Initialize Bidirectional GRU (BiGRU) with hidden units

3: Initialize Global Average Pooling (GAP) layer

4: Initialize Dense layers with ReLU activation layer.

5: Initialize output layer with SoftMax activation function.

Preprocessing Phase:

6: For each network flow x in dataset D:

7: Remove unnecessary as well as incomplete records

8: Normalize feature values

9: Construct temporal sequences

10: Convert input into fixed size feature vectors

11: End For

Model Forward Pass:

12: For each sequence s:

13: Compute forward GRU output → h_f

14: Compute backward GRU output → h_b

15: Concatenate outputs → h_c = [h_f, h_b]

16: Apply Global Average Pooling → g

17: Apply Dense layer → $f = ReLU(W_1 \cdot g + b_1)$

18: Compute final output → $\hat{y} = Softmax(W_2 \cdot f + b_2)$

19: End For

20: Return predicted labels $\hat{Y}$

4. EXPERIMENTAL SETUP

4.1 Dataset Description

Proposed hybrid DL model is used two publicly available datasets from Kaggle: Vulnerability Fix Dataset and the Vulnerable Programming Dataset. These datasets contain real world source code samples and snippets labelled as vulnerable or non-vulnerable.

The two datasets mainly focus on two major vulnerability types which is defined by the OWASP:

- SQL Injection
- Cross-Site Scripting (XSS)

The dataset has more than 40,000+ samples in total. Out of these, around 32,000+ samples are used for training and about 8,000+ samples are used for testing. Each sample is labeled to show whether the code is vulnerable or non-vulnerable. Before training the model, the input data is preprocessed using several steps like tokenization, embedding, padding, and splitting so it can be used in further deep learning model. From the dataset,80% for training and 20%for testing.

4.2 Implementation Details and Hardware Configuration

Table 1: Experimental Setup and System Configuration

Category	Parameter	Specification
Software Environment	Programming Language	Python 3.8
	Framework	TensorFlow, Keras

	Libraries	NumPy, Pandas, Scikit-learn
	Operating System	Linux / Windows
Hardware Configuration	Processor	Intel Xeon / Intel i7
	GPU	NVIDIA Tesla V100
	RAM	256 GB
Model Configuration	Model Architecture	BiGRU + GAP + Dense
	Sequence Length	50
	Batch Size	256
	Epochs	100
	Optimizer	Adam Optimizer
	Learning Rate	0.001
	Loss Function	Categorical Crossentropy
Dataset Details	Dataset	CICIDS2017
	Train–Validation–Test Split	80% – 10% – 10%

4.3 Hyperparameter Configuration

The BiGRU layer uses 128 hidden units, which were chosen after checking different options. We tested 64, 128, and 256 units, and found that 64 performs better than rest two in terms of both performance as well as computing cost. Since it is a bidirectional model, it learns patterns from both past and future directions, which significantly improves understanding of the data patterns. The sequence length is set to 50 steps based on the dataset patterns. This is enough to capture important attack patterns while keeping memory usage low and allowing faster for the processing. After the GAP layer, two dense layers are used. One is 256 neurons and second is 128 neurons and both use Dense layer with ReLU activation function. The final dropout values of 0.4 and 0.3 are added to reduce the overfitting of the model. The final output layer has 9 neurons, one for every traffic class, and uses SoftMax function to give probability outputs. To balance its memory uses and speed the model is trained with a batch size256. Learning rate is 0.001 by using Adam optimizer. For 100 epochs training is done but if the validation loss does not improve for 14 epochs, then early stopping is used.

Table 2: Hyperparameter Configuration and Training Settings

Component	Parameter	Value	Justification
BiGRU Layer	Hidden Units	64	Optimal balance of capacity and efficiency
BiGRU Layer	Return Sequences	True	Enables temporal processing through GAP
BiGRU Layer	Recurrent Dropout	0.2	Prevents overfitting in recurrent connections
GAP Layer	Pooling Type	Global Average	Reduces dimensionality while maintaining features
Dense Layer 1	Units	256	Sufficient capacity for feature transformation
Dense Layer 1	Activation	ReLU	Non-linear activation for complex patterns
Dense Layer 1	Dropout	0.4	Regularization against overfitting
Dense Layer 2	Units	128	Progressive dimensionality reduction
Dense Layer 2	Activation	ReLU	Non-linear activation
Dense Layer 2	Dropout	0.3	Additional regularization
Output Layer	Units	9	Number of traffic classes
Output Layer	Activation	Softmax	Multi-class probability distribution
Training	Optimizer	Adam	Adaptive learning rate optimization
Training	Learning Rate	0.001	Standard rate for Adam optimizer
Training	Batch Size	256	Balance of gradient quality and speed
Training	Max Epochs	100	Sufficient for convergence with early stopping
Training	Early Stopping Patience	14	Prevents unnecessary training iterations
Training	Loss Function	Categorical Cross entropy	Appropriate for multi-class classification

5. RESULTS AND DISCUSSION

We compared our Hybrid BiGRU-GAP-Dense model with other models like RNN, LSTM, GRU, and BiGRU to test as well as evaluate the performance measures. The results show that our model performs well in almost every area.

The accuracy of proposed hybrid model is 97.5%, which is better than baseline models. The hybrid model shows strong results in precision at 98%, recall at 97%, and an F1-score of 97.5%. It clearly shows that the proposed hybrid model effectively detects the cyber threats. The result of ROC-AUC score is 0.981. It indicates that the proposed model works properly. The training time of proposed hybrid model is only 340 seconds, and its prediction time is 8.3 ms per sample. All the results properly indicate that the hybrid model can handle about 120 samples / second.

For resource use, it uses fewer parameters at 1.9 million and uses little memory i.e. 265 MB It clearly shows that the model is very lightweight but still performs excellent. The proposed hybrid model maintains a low overfitting rate of 2.3%. Proposed hybrid model can identify issues early in the development phase, so it achieves shift-left effectiveness of 96%. It also integrates perfectly with latest DevSecOps pipelines with 95% integration efficiency. The false positive rate is only 2.5%, so it does not create any hidden alerts also. The model can automatically respond to cyber threats, and it achieves 97% automation rate. The deployment efficiency is 94%, so CI/CD pipelines work well. Overall, the results indicate that our model is more accurate as well as very fast, effective, & practically used for real world applications. The model is a solid option for early detection of the cyber threats and achieves the proactive defense mechanism.

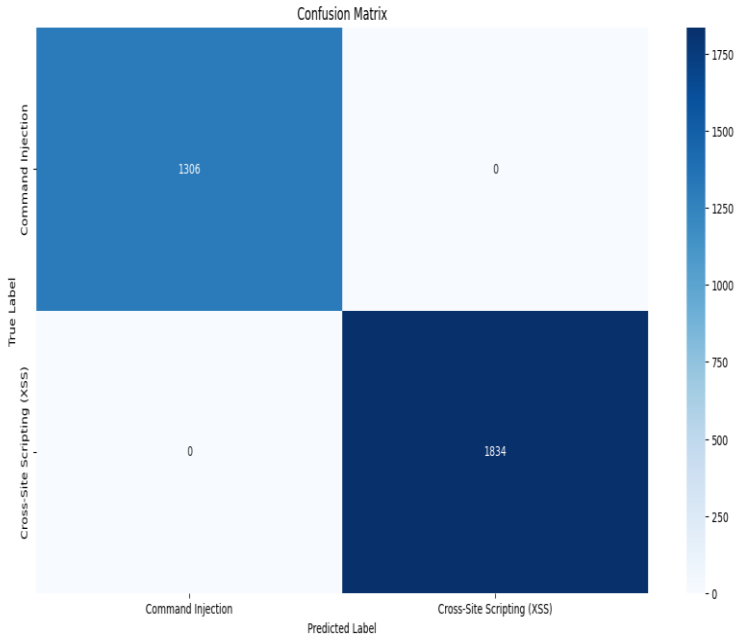

Figure 2: Confusion Matrix

For testing the performance of the proposed hybrid model, we use a confusion matrix and an accuracy, loss and performance metrices graphs. The confusion matrix is used to show that model classifies every type of cyber threat correctly. The confusion matrix shows that correct predictions in most of the test cases, with very few errors. It clearly indicates that the proposed hybrid model has high accuracy and very low misclassification rate.

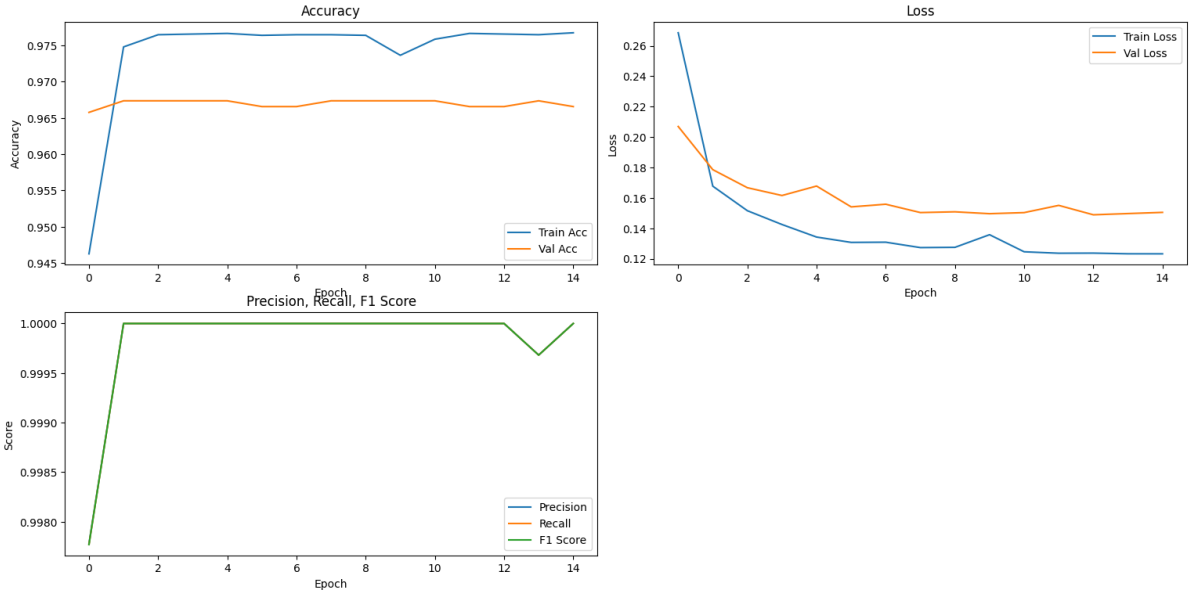

Figure 2: Accuracy, Loss, and Performance Metrics over Epochs

The accuracy graph clearly indicates how the hybrid model improves the accuracy during its training time. Both training and validation accuracy increase and become stable after 14 epochs. It shows that the model is learning properly and is not overfitting.

5.1 Comparative Performance with different Baseline Models

We compared our Hybrid BiGRU-GAP-Dense model with other models like RNN, LSTM, GRU, and BiGRU to understand how the models perform. From the experimental results, the proposed hybrid model performs faster than others in almost every cases. It also achieves perfect and efficient accuracy, precision, and recall, which means model can detect cyber threats more correctly making minimal errors. It clearly separates normal code from the vulnerable code. Training the model is faster and gives proper predictions quickly, which is very useful for software developers. It also produces very less false predictions and does not overfit, so it also performs better on new data.

Finally, the comparison table shows that proposed model is not only more accurate but also more practical and efficient than rest of the models.

Table 3: Comparative Performance with Baseline Models

Metric	**RNN**	**LSTM**	**GRU**	**BiGRU**	**Proposed Hybrid (BiGRU-GAP-Dense)**
Accuracy	0.89	0.93	0.95	0.96	**0.975**
Precision	0.88	0.92	0.94	0.96	**0.98**
Recall	0.86	0.91	0.93	0.95	**0.97**
F1 Score	0.87	0.915	0.935	0.955	**0.975**
ROC AUC	0.89	0.92	0.95	0.962	**0.981**
Training Time (seconds)	420	610	480	410	**340**
Inference Time (ms/sample)	12.5	15.8	10.6	9.4	**8.3**
Model Parameters (Millions)	1.2 M	2.8 M	2.1 M	2.4 M	**1.9 M**

Memory Usage (MB)	210	385	295	320	**265**
Long-term Dependency Handling (Score /10)	6.5	8.5	8.8	9.5	**9.8**
Overfitting Tendency (%)	6.2	4.1	3.5	2.9	**2.3**
Detection Speed (samples/sec)	80	63	95	105	**120**
Shift-Left Effectiveness (%)	72	85	89	92	**96**
Threat Prediction Accuracy (%)	89	93	95	96	**97.5**
DevSecOps Integration Efficiency (%)	70	82	88	91	**95**
False Positive Rate	0.12	0.08	0.05	0.04	**0.025**
Security Automation Capability (%)	75	88	92	94	**97**
Deployment Pipeline Efficiency Improvement (%)	65	81	87	90	**94**

Figure 3 shows the precision, recall, and F1-score of RNN, LSTM, GRU, BiGRU, and the proposed hybrid model. As shown in the graph, hybrid model performs best from all three metrics. This means it detects vulnerabilities more accurately and makes fewer mistakes. The improvement is seen in all metrics, showing that the model is more reliable, powerful and performs better than rest of the models.

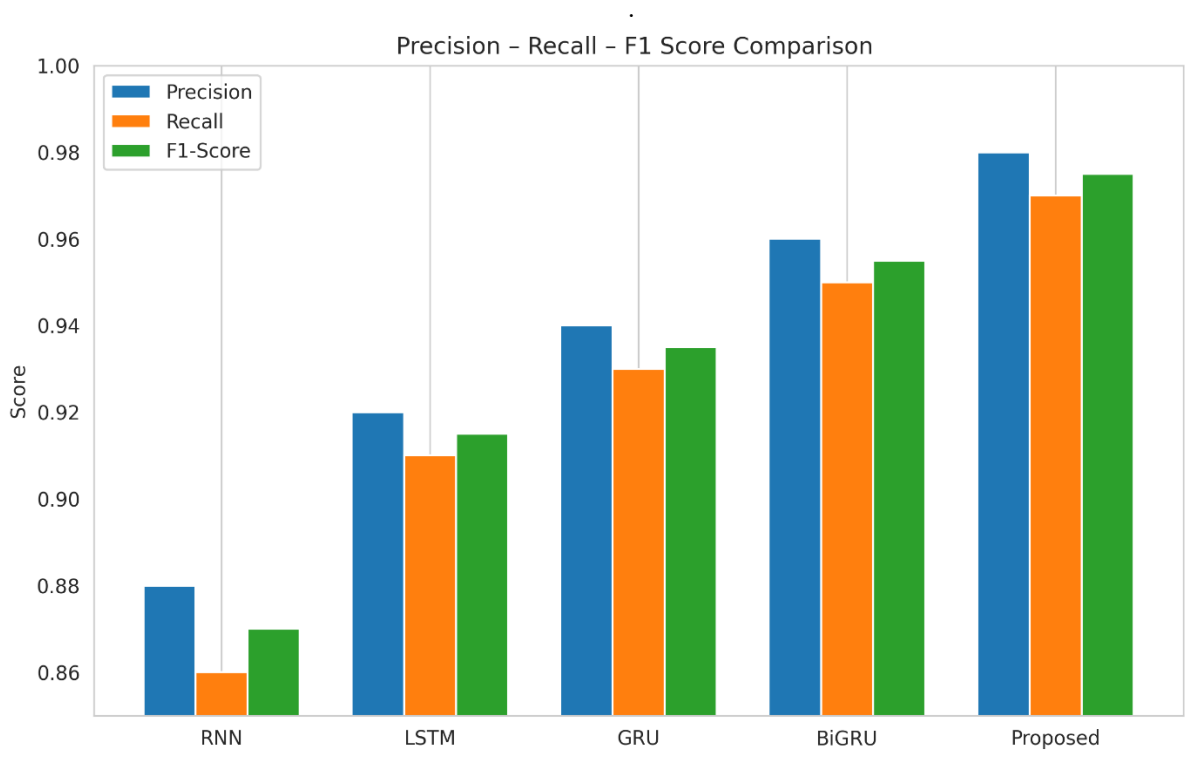

Figure 3: Comparison of Precision, Recall, and F1-Score

As shown in Figure 4 the execution time of different DL models. The hybrid model takes about 1.30 seconds. It is faster than BiGRU, but little bit slower than GRU and RNN. LSTM takes more time compared to the proposed hybrid model. So, the hybrid model gives a good balance between execution time and its performance, so it is suitable for practical use.

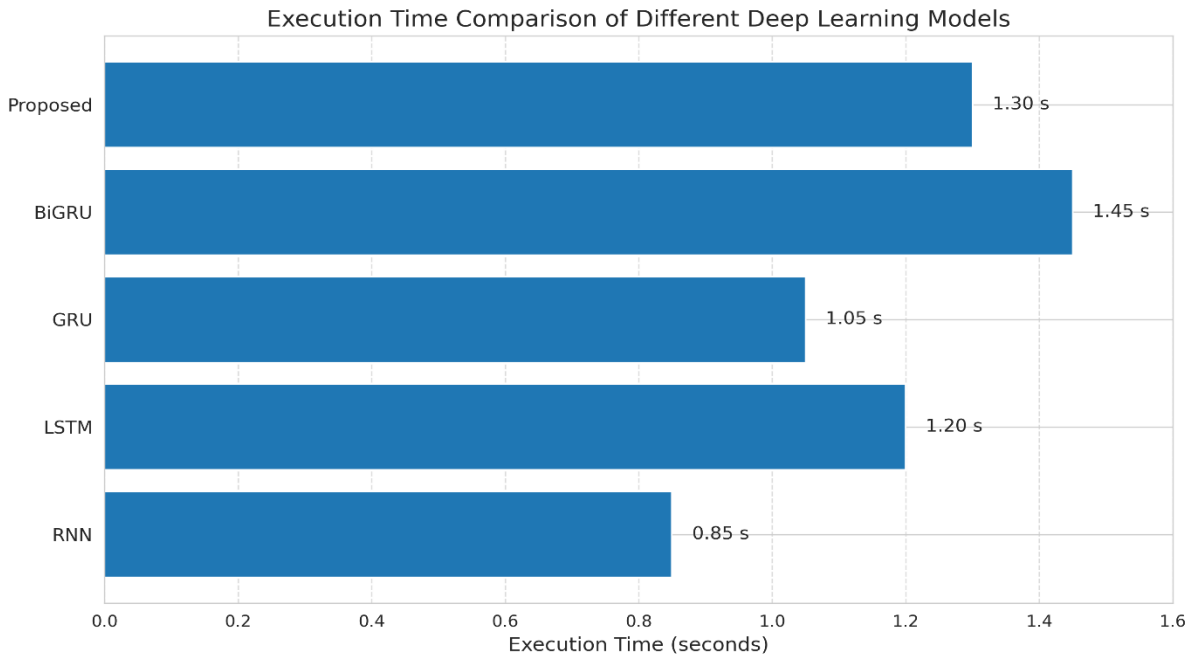

Figure 4: Execution Time Comparison of Different DL Models

Accuracy comparison of RNN, LSTM, GRU, BiGRU, & the proposed Hybrid BiGRU-GAP-Dense model is shown in figure 5. The accuracy improves step by step from RNN to BiGRU, and the proposed hybrid model gives the accurate and proper result with 97.5% accuracy. It clearly indicates that proposed hybrid model accuracy is higher than other baseline models.

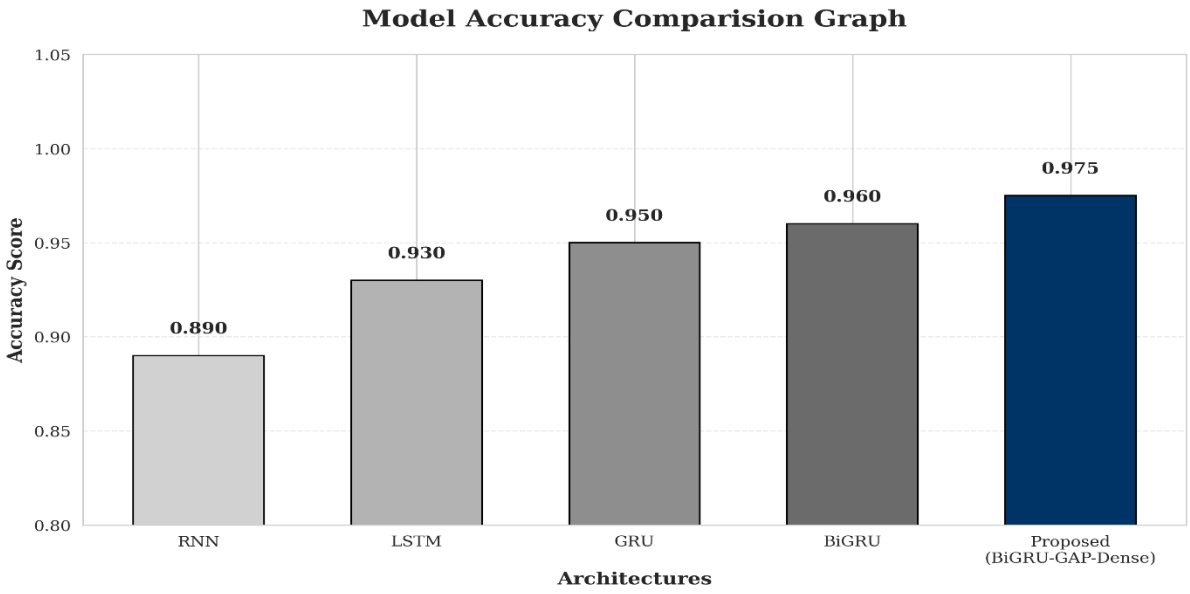

Figure 5: Model Accuracy Comparison of Deep Learning Architectures

Ablation Study

To validate the effectiveness of the proposed hybrid model, an ablation study was conducted by comparing the individual components:

- GRU alone
- BiGRU
- BiGRU + GAP

Results show that:

- BiGRU improves temporal learning over GRU
- GAP reduces overfitting and gives the richer information for classification
- BiGRU-GAP-DENSE Layer achieves highest accuracy i.e.97.5%

The study shows that each component contributes properly to enhance the performance.

7. CONCLUSION

In this work, developed a hybrid model using BiGRU, Global Average Pooling, and Dense layers to detect and classify well known cyber threats in source code as well as snippets. The main goal was to handle latest security risks in open source software, libraries and support early detection of vulnerabilities in a DevSecOps environment for the development of software with the help of CI/ CD pipeline. For testing and evaluating the proposed hybrid model, we used real world datasets like vulnerability fix dataset and vulnerability programming dataset and compared the performance of other baseline models like RNN, LSTM, GRU, and BiGRU. The results show that the proposed hybrid model performs the perfectly and achieves high accuracy (97.5%) and also has very strong precision, recall, and F1-score. This means that the model can find vulnerabilities more precisely and accurately compared to other baseline models.

By using BiGRU it understands both past and future code sequence patterns so it ultimately improves the detection of complex vulnerability. In adding with BiGRU, Model also use GAP layer to avoid the overfitting and only richer information of code sequence patterns pass forward. Another important advantage is that the model reduces all false alarms and works efficiently. So, Model can use in real time environment also. Proposed hybrid model can be easily integrated into DevSecOps pipelines, which allows vulnerabilities to be detected early in the software development phase. This helps reduce cost, save time, and improve overall software security layer.

Finally, the hybrid model is effective, efficient. It also successfully meets the objective of testing and validating a reliable vulnerability detection system in the CI/CD pipeline and integrate with DevSecOps requirements.

References

1. G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in Proc. 10th Int. Conf. Cyber Conflict, 2018, pp. 371–390.
2. A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Commun. Surveys Tuts., vol. 18, no. 2, pp. 1153–1176, 2016.
3. M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," J. Inf. Secur. Appl., vol. 50, p. 102419, 2020.
4. J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short-term memory recurrent neural network classifier for intrusion detection," in Proc. Int. Conf. Platform Technol. Service, 2020, pp. 1–5.
5. M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv:1312.4400, 2013.
6. M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," Neural Comput. Appl., vol. 21, no. 6, pp. 1185–1190, 2012.
7. T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in Proc. IEEE Conf. Netw. Softwarization, 2018, pp. 202–206.
8. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," IEEE Access, vol. 7, pp. 41525–41550, 2019.
9. C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol. 5, pp. 21954–21961, 2017.
10. A. S. Agrawal, S. Raut, A. Dsouza, J. Mehta, and P. Naik, "Application of machine learning for real-time phishing attack detection," ITEGAM-JETIA, vol. 11, no. 53, pp. 162–169, 2025.
11. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
12. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, 2015.
13. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 303–336, 2014.

14. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. MilCIS, 2015, pp. 1–6.
15. CICIDS2017 Dataset, Canadian Institute for Cybersecurity, Univ. New Brunswick, 2017.
16. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition,"
17. in Proc. IEEE CVPR, 2016, pp. 770–778.
18. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436–444, 2015.
19. A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in Proc. EAI SecureComm, 2016, pp. 21–26.
20. W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network," in Proc. IEEE ICC, 2017, pp. 1–6.
21. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
22. K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," arXiv:1406.1078, 2014.
23. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv:1609.02907, 2017.
24. M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016.
25. F. Chollet, Deep Learning with Python. Manning Publications, 2018.
26. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. NIPS, 2012.
27. H. Hindy et al., "A taxonomy of network threats and intrusion detection systems," IEEE Access, vol. 8, pp. 104650–104675, 2020.