Journal of Data Analysis and Information Processing, 2023, 11, 371-387
https://www.scirp.org/journal/jdaip
ISSN Online: 2327-7203
ISSN Print: 2327-7211

A Hybrid DNN-RBFNN Model for Intrusion Detection System

Wafula Maurice Oboya*, Anthony Waititu Gichuhi, Anthony Wanjoya

Department of Statistics and Actuarial Sciences, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya
Email: *oboyamaurice@gmail.com

How to cite this paper: Oboya, W.M., Gichuhi, A.W. and Wanjoya, A. (2023) A Hybrid DNN-RBFNN Model for Intrusion Detection System. *Journal of Data Analysis and Information Processing*, **11**, 371-387.
https://doi.org/10.4236/jdaip.2023.114019

Received: August 19, 2023
Accepted: October 29, 2023
Published: November 1, 2023

Abstract

Intrusion Detection Systems (IDS) are pivotal in safeguarding computer networks from malicious activities. This study presents a novel approach by proposing a Hybrid Dense Neural Network-Radial Basis Function Neural Network (DNN-RBFNN) architecture to enhance the accuracy and efficiency of IDS. The hybrid model synergizes the strengths of both dense learning and radial basis function networks, aiming to address the limitations of traditional IDS techniques in classifying packets that could result in Remote-to-local (R2L), Denial of Service (Dos), and User-to-root (U2R) intrusions.

Keywords

Dense Neural Network (DNN), Radial Basis Function Neural Network (RBFNN), Intrusion Detection System (IDS), Denial of Service (DoS), Remote to Local (R2L), User-to-Root (U2R)

1. Introduction

In recent years, computer networks have increasingly become relied upon for communication and data exchange. This reliance has resulted in a rise in cyber-attacks and malicious activities. The effectiveness of traditional security measures, such as firewalls and intrusion detection systems, is diminishing when faced with current security threats. The continuously evolving nature of malicious network activity requires organizations to adopt advanced security measures to ensure adequate protection. Therefore, developing effective and efficient methods for real-time detection and mitigation of these malicious activities is crucial. To classify these malicious network activities, we can utilize a hybrid of a Dense Radial Basis Function Neural Network (DRBFNN). In this approach, we input data into a Dense Neural Network (DNN). Subsequently, the output feeds into a Radial Basis Function Neural Network (RBFNN). The DNN functions as the initial stage of the network architecture, taking responsibility for processing and extracting features from the input data. It consists of multiple layers of interconnected nodes, known as neurons, which use nonlinear activation functions to transform the input data and learn complex representations. We then pass the output from the DNN as input to the RBFNN. The RBFNN is a type of neural network that employs radial basis functions as its activation functions [1]. These functions assess the distance between the input data and a set of predefined center points, and they combine their outputs to generate the final predictions or outputs of the network. By merging the capabilities of both the DNN and RBFNN, we aim to utilize the feature extraction and representation learning capabilities of the DNN, along with the RBFNN's ability to model complex patterns using radial basis functions. This hybrid approach can potentially improve the network's performance across a range of tasks, including classification, regression, and pattern recognition. It achieves this by effectively capturing and processing the input data.

2. Background

2.1. Intrusion Detection Systems

In the early stages of computer network development, experts manually analyzed system logs and access records as the initial methods for intrusion detection. However, as cyber-attack techniques grew more intricate, the need for automated and advanced solutions became apparent. From 1984 to 1986, Dorothy Denning and Peter Neumann pioneered a prototype model for an Intrusion Detection System (IDS) known as the Intrusion Detection Expert System (IDES). **Figure 1** visually represents the implementation of IDS within systems.

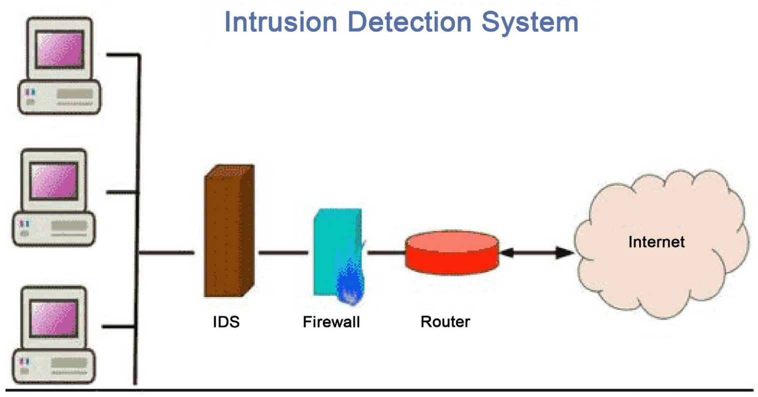


Figure 1. Intrusion detection system (source: comodo.com).

Over the subsequent decades, signature-based detection gained prominence due to the recognition of established attack patterns and the creation of rule-based detection engines. Notable contributions in this area include a comprehensive survey on secure networks conducted by [2] and the subsequent launch of Snort, an open source Network-based Intrusion Detection System (NIDS) that employed signature matching for real-time identification of attacks, as presented in the paper [3]. While this method proved effective against known threats, it faced challenges in dealing with zero-day attacks and maintaining an up-to-date signature database.

2.2. IDS in DNN and RBFNN

Studies on the detection of malicious network activities have been plentiful, but there has been limited research on the use of a hybrid models classification approach for this purpose. Parul and Gurjwar employed a layered approach using the Decision Tree classifier to train their IDS [4]. The approach provided good results for each layer. In addition, the Random Forest algorithm was utilized and showed generally good performance, but had limited ability to detect U2R (user-to-root) attacks with a low rate of classification. The authors recommended modifications to the Random Forest algorithm to improve the U2R layer results. The IDS was evaluated using the KDDcup99 dataset and demonstrated improvement compared to the newer NSL KDD dataset.

A Hybrid DNN-RBFNN Model for the classification of malicious network activity can build upon the existing research by incorporating additional algorithms and techniques, such as Dense Neural Network (DNN) [5] along with Radial Basis Function Neural Network (RBFNN), to enhance the accuracy and efficiency of malicious network activity classification. One significant advantage of dense neural networks in IDS is their ability to learn complex patterns and relationships from large-scale network data. Zarai, R. *et al.* [6] conducted an experimental study and reported that a dense neural network architecture achieved an accuracy of 94% in detecting various types of network intrusions, outperforming traditional rule—based IDS approaches. Furthermore, the application of deep learning techniques, including the combination of convolutional neural networks (CNN) and dense neural networks, has demonstrated encouraging outcomes in Intrusion Detection Systems (IDS).

In a study conducted by [7], various deep learning architectures were compared, and it was discovered that a model that employed CNN for feature extraction and a dense neural network for classification exhibited superior performance in detecting advanced persistent threats (APTs). Just like any other neural network model, the vulnerability of DNNs to adversarial attacks poses a significant concern, prompting researchers to propose various techniques and strategies for mitigating these attacks. Promising approaches include adversarial training, the utilization of defense mechanisms, and input transformations. Adversarial training, as outlined in the paper [8], involves augmenting the training dataset with adversarial examples, compelling the DNN to develop increased resilience against attacks.

Radial Basis Function (RBF) networks have been utilized in the field of intrusion detection systems (IDSs) for tasks such as determining the normality or anomaly of network packets [9]. These artificial neural networks typically comprise three layers: an input layer, a hidden layer, and an output layer. The hidden layer consists of radial basis functions (RBFs), mathematical functions that as-sign a value to an input based on its radial distance from a central point. The RBFs in the hidden layer are trained using unsupervised learning methods to establish a mapping from the input space to the output space. **Figure 2** illustrates the fundamental structure of a Radial Basis Function Neural Network.

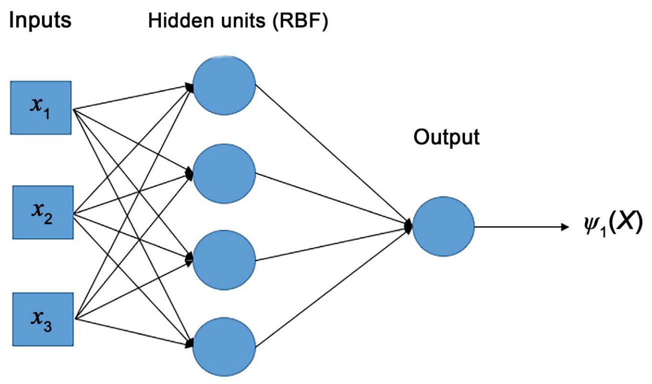


Figure 2. Radial basis function neural network © [Gildardo Sanchez-Ante].

2.3. Hybrid DNN-RBFNN in Intrusion Detection

Research by [10] emphasizes that signature-based detection remains effective in identifying well-known attacks with established patterns. This approach has proven successful in countering attacks that have recognizable footprints, ensuring rapid response to known threats. Furthermore, anomaly-based detection, as highlighted by [11], is adept at detecting novel or zero-day attacks that lack predefined signatures. By establishing a baseline of normal network behavior, any deviations from this baseline can be flagged as potential intrusions.

Despite their strengths, current IDS face notable limitations. Signal-based detection is ineffective against zero-day attacks, as they do not have previously identified patterns. Similarly, anomaly-based detection often struggles with false positives and negatives due to its reliance on defining "normal" behavior, which can vary widely in complex networks. Signature-based detection can be enhanced by leveraging the DNN component of the hybrid model. DNNs are well-suited for learning complex patterns and features from raw data. They can be trained on a vast amount of historical network traffic to identify subtle variations in attack patterns, even when attackers slightly modify them to evade traditional signature-based detection. Anomaly-based detection can benefit from the RBFNN component. RBFNNs are proficient in capturing deviations from established norms, effectively detecting novel or zero-day attacks. By using the RBFNN's ability to identify anomalies, the hybrid model can better identify new and unknown attack patterns that lack predefined signatures.

3. Methodology

3.1. Data Preprocessing

Data preprocessing plays a pivotal role in achieving high predictive performance with machine learning models. Hybrid models, which combine different types of neural networks, have shown promise in tackling complex tasks by leveraging the strengths of each component. The combination of Dense Neural Networks (DNNs) and Radial Basis Function Neural Networks (RBFNNs) offers a powerful approach to capture both global and local patterns in data. However, the effectiveness of these hybrid models heavily relies on the quality of the input data.

3.2. Data Normalization

Data normalization ensures uniform scaling of input features, a critical step to facilitate the learning process. For hybrid IDS models, data normalization enhances convergence speed and minimizes the dominance of certain features. Common techniques such as Z-score normalization and Min-Max scaling are applied to bring the features to a common scale [12].

3.3. Feature Selection

Feature selection is instrumental in eliminating irrelevant or redundant attributes that could lead to overfitting or increased computational complexity. Hybrid IDS models benefit from a subset of features that collectively capture both global and local patterns. Methods like mutual information, Recursive Feature Elimination (RFE), and tree-based techniques are employed to ensure effective feature selection [13].

3.4. Developed Hybrid Model

The developed hybrid DNN-RBFNN model uses both DNN and RBFNN for the classification of intrusion detection. The DNN was designed with several hidden layers to learn the complex feature representation of the input data, on the other hand, RBFNN, was used to perform the final classification. Mathematically the hybrid Dense Radial Basis Function Neural Network can be represented as:

$$\text{net}_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_j^h \tag{1}$$

where w_{ji}^{h} is the weight on the connection from the ith input unit, and θ_j^h is the bias term. The "h" superscript refers to quantities on the hidden layer. if the activation of this node is equal to the net input; then, the output of this node is

$$i_{pj} = f_j^h\left(\text{net}_{pj}^{h}\right) \tag{2}$$

The equations for the output nodes are

$$\text{net}_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} + \theta_k^o \tag{3}$$

$$o_{pk} = f_k^o\left(\text{net}_{pk}^{o}\right) \tag{4}$$

where the "o" superscript refers to quantities on the output layer. The obtained outputs, O_{pk}, of the Dense Neural Network layers are passed to the Radial Basis Function Neural Network layer which is responsible for producing the final output of the hybrid model.

$$\phi_i\left(O_{pk}\right)=\exp\left[-\frac{\left(O_{pk}-\mu_i\right)^{\mathrm{T}}\left(O_{pk}-\mu_i\right)}{2\sigma_i^2}\right],\quad i=1,2,\cdots,K \tag{5}$$

The ultimate output of the neural network is determined through a process of linear weighted summation involving the outputs derived from the hidden layer.

$$F\left(x_i\right)=w_i^{\mathrm{T}}\phi_i\left(O_{pk}\right) \tag{6}$$

3.4.1. Parameter Estimates of the Developed Model

Once we've obtained the final output function we then calculate the cost function to determine the error between the predicted value and the original values. To achieve this, we provide an input x for which we know the corresponding output $f\left(x_i\right)$ to the network and observe the computed result. In a broader context, we can assess the network's performance on a testing set $\left(x_1,f\left(x_1\right)\right),\left(x_2,f\left(x_2\right)\right),\cdots,\left(x_n,f\left(x_n\right)\right)$, where each $\left(x_i,f\left(x_i\right)\right)$ represents an input-output pair. By comparing the computed outputs of the network with the true values from the testing set, we can evaluate how well the network performs.

$$C:=\frac{1}{2N}\sum_{i=1}^{N}\left\|f\left(x_i\right)-F\left(x_i\right)\right\|^2, \tag{7}$$

By defining $C_x:=\frac{1}{2N}\left\|f\left(x\right)-F\left(x\right)\right\|^2$, we can observe that this function becomes large when our network poorly approximates *f*, and small when the approximation is accurate.

$$C=\sum_{i=1}^{N}C_{x_i} \tag{8}$$

After we've obtained the cost function we then back propagate the hybrid network model updating the weights,

$$\nabla C=\nabla\left(\sum_{i=1}^{N}C_{x_i}\right)=\sum_{i=1}^{N}\nabla C_{x_i} \tag{9}$$

which implies that we can perform this process for each data point individually and then accumulate the gradients by adding their values together.

In Equation (8), we introduced the notation i_{pj} to represent the vector that captures the activations of the nodes in the *j*th layer of the network. However, for the context, it will be advantageous to examine the values propagated from the preceding layer prior to the application of the activation function.

$$z_j^l:=\sum_k w_{j,k}^l x_k^{l-1}+b_j^l \text{ so that } a_j^l=\sigma\left(z_j^l\right) \text{ and } i_{pj}^l=\sigma\left(Z^l\right) \tag{10}$$

In the preceding equation, we denote Z_l as $Z^l:=\sum_k z_k^l e_k \quad e_k$, where each entry corresponds to the values z_j^l (e_i are the standard basis vectors). Moreover, examining the quantity.

$$\delta_j^l:=\frac{\partial C}{\partial z_j^l} \quad \text{and} \quad \Delta^l:=\sum_k \delta_k^l e_k \tag{11}$$

These values hold significance in the backward propagation of the algorithm through the network and have a direct relationship with $\frac{\partial C}{\partial w_{i,j}^{l}}$ and $\frac{\partial C}{\partial b_{j}^{l}}$ through the chain rule, as

$$\frac{\partial C}{\partial w_{i,j}^{l}} = \frac{\partial C}{\partial z_{j}^{l}}\frac{\partial z_{j}^{l}}{\partial w_{i,j}^{l}} = \delta_{j}^{l}a_{i}^{l-1} \quad \text{and} \quad \frac{\partial C}{\partial b_{j}^{l}} = \frac{\partial C}{\partial z_{j}^{l}}\frac{\partial z_{j}^{l}}{\partial b_{j}^{l}} = \delta_{j}^{l}. \tag{12}$$

If we can compute the values of δ_{j}^{l}, we will have successfully obtained our gradient, given that a_{j}^{l-1} is readily available for any node in the network. Our initial step is to calculate this value for the last layer of the network, denoted as δ_{j}^{L} for a network with L layers. By observing that $a_{j}^{L} = \sigma\left(z_{j}^{L}\right)$, we can apply the chain rule once again to deduce that

$$\delta_{j}^{L} = \frac{\partial C}{\partial a_{j}^{L}}\frac{\partial a_{j}^{L}}{\partial z_{j}^{l}} = \frac{\partial C}{\partial a_{j}^{L}}\sigma'\left(z_{j}^{L}\right) \tag{13}$$

$$\delta_{j}^{L} = \left(a_{j}^{L} - y_{j}\right)\sigma'\left(z_{j}^{L}\right) \tag{14}$$

To obtain δ_{j}^{L-1}, we can further use chain rule to propagate this value backward in the network.

$$\delta_{j}^{L-1} = \frac{\partial C}{\partial z_{j}^{L-1}} = \nabla_{Z^{L}}C \cdot \frac{\partial Z^{L}}{\partial z_{j}^{L-1}} = \sum_{i}^{k}\frac{\partial C}{\partial z_{i}^{L}}\frac{\partial z_{i}^{L}}{\partial z_{j}^{L-1}} = \sum_{i}^{k}\delta_{i}^{L}\frac{\partial z_{i}^{L}}{\partial z_{j}^{L-1}} \tag{15}$$

Once more, focusing on the term $\frac{\partial z_{i}^{L}}{\partial z_{j}^{L-1}}$, we can determine that

$$\begin{aligned}\frac{\partial z_{i}^{L}}{\partial z_{j}^{L-1}} &= \frac{\partial\left(\sum_{k} w_{i,k}^{L}a_{k}^{L-1} + b_{i}^{L}\right)}{\partial z_{j}^{L-1}} = \frac{\partial\left(\sum_{k} w_{i,k}^{L}\sigma\left(z_{k}^{L-1}\right)^{+} b_{i}^{L}\right)}{\partial z_{j}^{L-1}} \\ &= \frac{\partial\left(w_{i,j}^{L}\sigma\left(z_{j}^{L-1}\right)\right)}{\partial z_{j}^{L-1}} = w_{i,j}^{L}\sigma'\left(z_{j}^{L-1}\right),\end{aligned} \tag{16}$$

Therefore,

$$\delta_{j}^{L-1} = \sum_{i}^{k}\delta_{i}^{L}w_{i,j}^{L}\sigma'\left(z_{j}^{L-1}\right) \tag{17}$$

The weights are adjusted until a stopping criterion is met.

3.4.2. Adversarial Test on the Developed Model

The developed hybrid DNN-RBFNN model classifier with a softmax output activation, denoted as $\tilde{y} = f\left(\theta, x\right)$, where θ represents the model parameters and $\left(x, y\right)$ represents an instance-label pair from the dataset. We generate an adversarial instance x' using the Fast Gradient Sign Method (FGSM), which aims to maximize the loss $L\left(x', y\right)$ while adhering to the l_{∞} perturbation constraint, $\left\|x' - x\right\|_{\infty} \le \varepsilon$, where ε represents the attack strength specific to the dataset.

Utilizing a first-order approximation, we can approximate the loss function as $L\left(x', y\right) \approx L\left(x, y\right) + \nabla_{x}L\left(x, y\right)^{\mathrm{T}} \cdot \left(x' - x\right)$. With this approximation, we can express the computation of the adversarial instance x' as follows:

$$x' = x + \varepsilon \cdot \operatorname{sign}\left(\nabla_x L(x, y)\right),$$

where $\operatorname{sign}\left(\nabla_x L(x, y)\right)$ represents the element-wise sign function applied to the gradient of the loss function with respect to *x*.

3.5. Evaluation Metrics

To assess the performance of the hybrid DNN-RBFNN model, appropriate evaluation metrics are selected. One such metric includes a confusion matrix in Table 1, which is a tabular representation that provides an overview of how well a classification model performs in predicting class labels when evaluated against test data.

Table 1. Confusion matrix.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) refers to the accurate identification of positive events among the observed data.

True Negative (TN) corresponds to the correct identification of negative events where both the actual and predicted values are negative.

False Positive (FP) denotes the situation where the predicted value is positive, but the actual value is negative.

False Negative (FN) refers to the scenario where the predicted value is negative, but the actual value is positive.

3.5.1. Specificity

In multi-class classification, specificity is used to evaluate the performance of a model for each individual class. It measures the ability of the model to correctly identify instances of a specific class while considering the other classes as negatives.

Specificity for a specific class can be defined as:

$$\text{Specificity}_C = \frac{\text{True Negatives}_C}{\text{True Negatives}_C + \text{False Positives}_C}$$

To calculate the specificity for a particular class, we treat that class as the negative class and compute the ratio of true negatives for that class to the sum of true negatives and false positives for that class.

3.5.2. Sensitivity

Sensitivity can be calculated for each individual class to evaluate the model's performance in correctly identifying instances of a specific class. The sensitivity for a specific class can be expressed as:

$$\text{Sensitivity}_C = \frac{\text{True Positives}_C}{\text{True Positives}_C + \text{False Negatives}_C} \tag{18}$$

3.5.3. Accuracy

Mathematically, overall accuracy is defined as the ratio of the total number of correctly classified instances (both true positives and true negatives) to the total number of instances in the dataset:

$$\text{Overall Accuracy} = \frac{\text{Number of Correctly Classified Instances}}{\text{Total Number of Instances}} \tag{19}$$

It provides a general assessment of how well the model performs across all classes.

4. Results and Discussions

The NSL KDD dataset is a widely recognized and valuable resource for studying intrusion detection in network security. It has become a standard benchmark dataset, building upon the original KDD Cup 1999 dataset and incorporating improvements. In this dataset, the target class represents different types of attacks that can occur in network security. These attacks can be categorized into either binary or multi-classification problems. The dataset consisted of 148,517 in-stances, which were classified into different types of intrusions: DoS (Denial of Service), Probe, U2R (User to Root), R2L (Remote to Local), and normal intrusions. Figure 3 shows the multi-class distribution of the target variables in the dependent variable class.

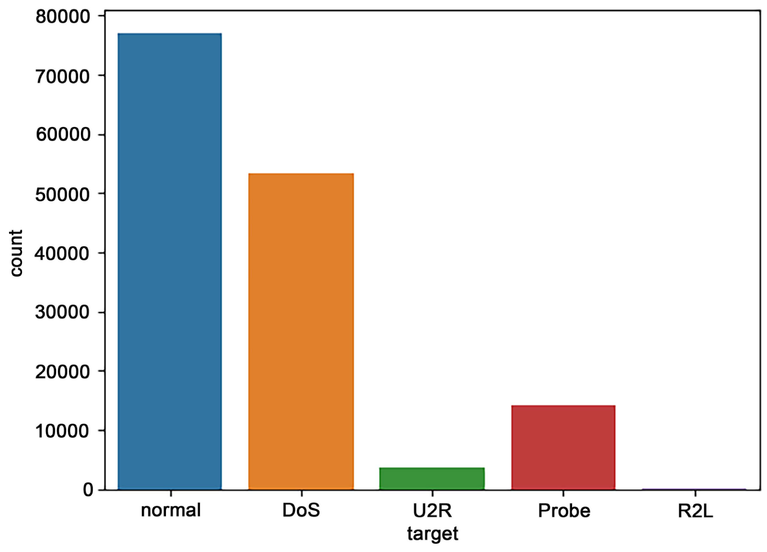


Figure 3. Count plot of target class distribution.

Among the 148,517 packets, the largest portion, encompassing 77,054 instances, represented normal intrusions. Denial of Service (DoS) at-tacks account for 53,385 instances, while Probe incidents were recorded at 14,077. User-to-root

(U2R) intrusions were relatively fewer, comprising 3749 instances, and Remote to Local (R2L) intrusions were the least common with only 252 instances.

4.1. Feature Selection

Feature selection is a crucial step in building robust and efficient machine-learning models. One popular algorithm for feature selection is the SelectKBest algorithm, which yielded the best results in terms of accuracy and computational time when used on various cybersecurity datasets [14]. SelectKBest algorithm assigns a scoring function that quantifies the relevance of each feature. Figure 4 represents the top 15 selected features using mutual information score statistics.

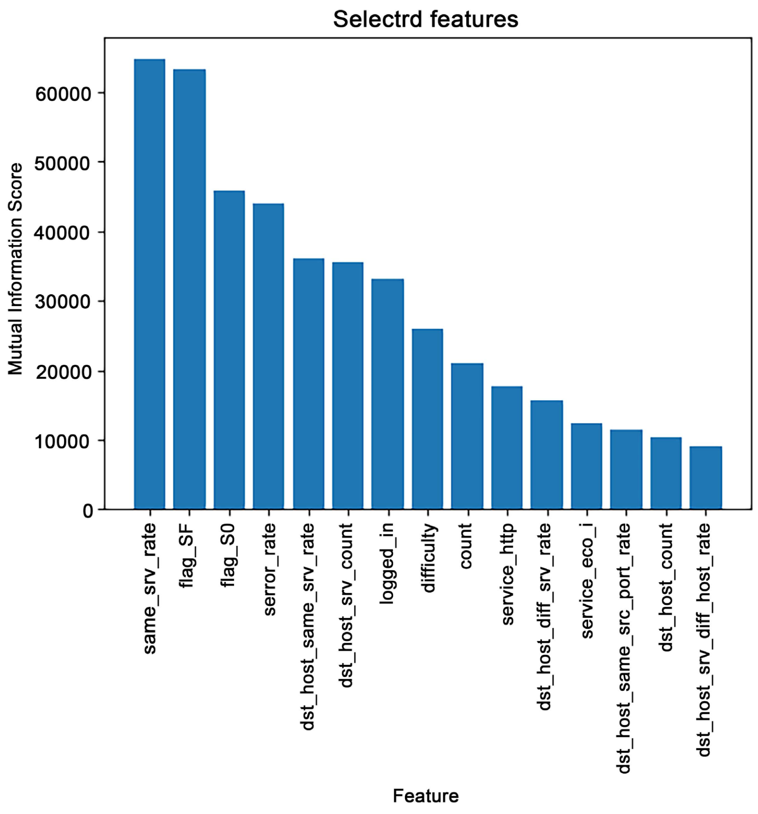
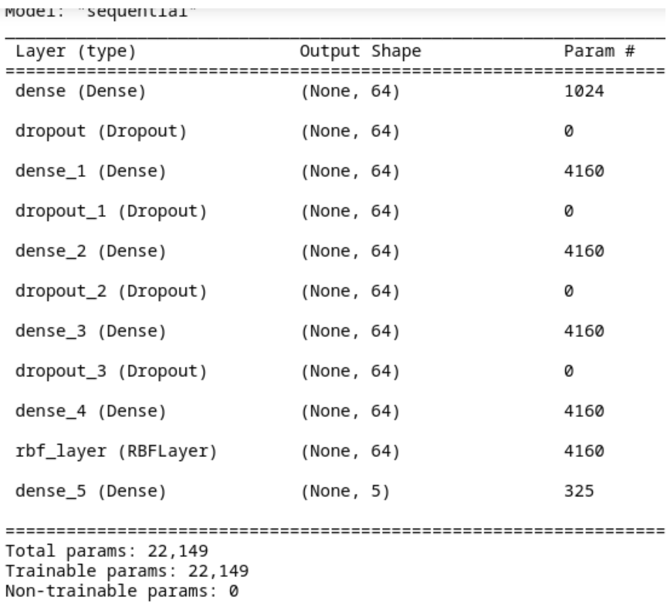


Figure 4. Top 15 selected features with the highest mutual info scores.

Where count: provides an indication of the connection rate, difficulty: measures the difficulty level of the network connection, dst_host_count:provides information about the connection rate to a specific destination, dst_host_diff_srv_rate: represents the rate of different services accessed on the destination host compared to the total number of connections made to that host, dst_host_same_src_port_rate: indicates the rate of connections originating from the same source port to a specific destination host, dst_host_same_srv_rate: represents the rate of connections to the same service on the destination host compared to the total number of connections made to that host, dst_host_srv_count: provides information about the service connection rate, dst_host_srv_diff_host_rate: measures the rate of connections to different hosts using the same service compared to the total number of connections to that service, flag_S0: if the connection's TCP flag is set to S0, it represents a connection attempt without synchronization, flag_SF: if the connection's TCP flag is set to SF, it represents a normal established connection, logged_in: It signifies whether the user is logged in or not, based on the connection, same_srv_rate: represents the rate of connections to the same service as the current connection, serror_rate: denotes the rate of connections that resulted in a TCP "SYN" error, service_eco_i: represents a specific network service or protocol, such as "eco_i" in this case and finally service_http: represents the HTTP network service, indicating connections related to web browsing and communication.

4.2. Developed Hybrid DNN-RBFNN Model

To create the hybrid model, we employed a two-step approach. First, we trained the Dense Radial Basis Function Neural Network (RBFNN) using the training dataset. The DNN portion of the model processed the input data and generated intermediate outputs. These outputs were then fed into the Radial Basis Function Neural Network, which further refined the representations and produced the final predictions. Figure 5 provides a visual representation of the developed model. It illustrates the interconnected layers and nodes of both the DNN and RBFNN components.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                1024

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 64)                4160

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 64)                4160

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 64)                4160

 dropout_3 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 64)                4160

 rbf_layer (RBFLayer)        (None, 64)                4160

 dense_5 (Dense)             (None, 5)                 325

=================================================================
Total params: 22,149
Trainable params: 22,149
Non-trainable params: 0
```

Figure 5. Developed hybrid model summary architecture.

4.3. Performance of the Developed Hybrid Model

Evaluating the performance of our developed hybrid model is crucial. In Figure 6, we can see the confusion matrix, which provides valuable insights into key metrics such as specificity, sensitivity, and accuracy specifically calculated for our hybrid model. These metrics help us understand how well our model is performing and make informed decisions based on the results.

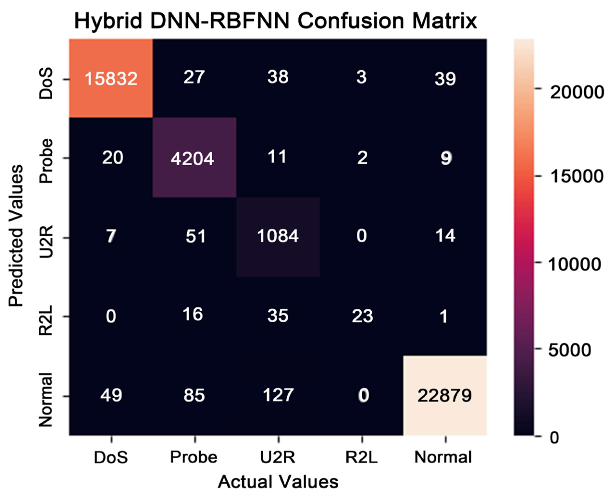


Figure 6. Adversarial model confusion matrix.

4.4. Adversarial Examples Test on the Developed Model

To thoroughly evaluate the resilience of the developed model against adversarial examples, we conducted a rigorous assessment using the FGSM attack. The FGSM technique, introduced by [15], is notorious for its malicious nature. It involves perturbing the input data by utilizing the gradient of the loss function for the input. By carefully modifying the data in the direction of the gradient, the FGSM attack aims to deceive the model.

This evaluation served as a comprehensive test to determine the model's ability to withstand adversarial attacks. Based on the findings presented in Figure 7, we observed that the developed model exhibited susceptibility to adversarial examples. It could be tricked into failing to detect certain packets that could potentially lead to User-to-Root (U2R) and Remote-to-Local (R2L) attacks.

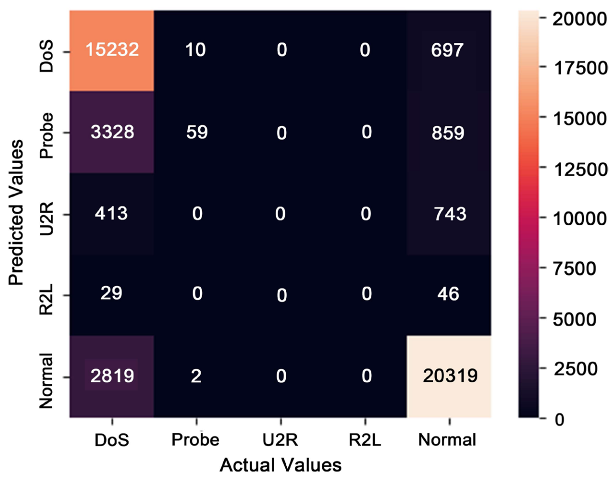


Figure 7. Adversarial examples confusion matrix.

4.5. Model Optimization

One challenge we face when training neural networks is deciding on the right number of training epochs to use. If we use too many epochs, the network can become too specialized and only perform well on the specific training data, which is called overfitting [16]. On the other hand, if we use too few epochs, the network may not fully grasp the patterns in the data and perform poorly, which is known as underfitting. So, finding the balance and determining the sweet spot for the number of training epochs is crucial for training a successful neural network. Figure 8 displays plots for the relationship between the number of epochs and the corresponding accuracy, providing valuable insights into the number of epochs each model takes to converge.

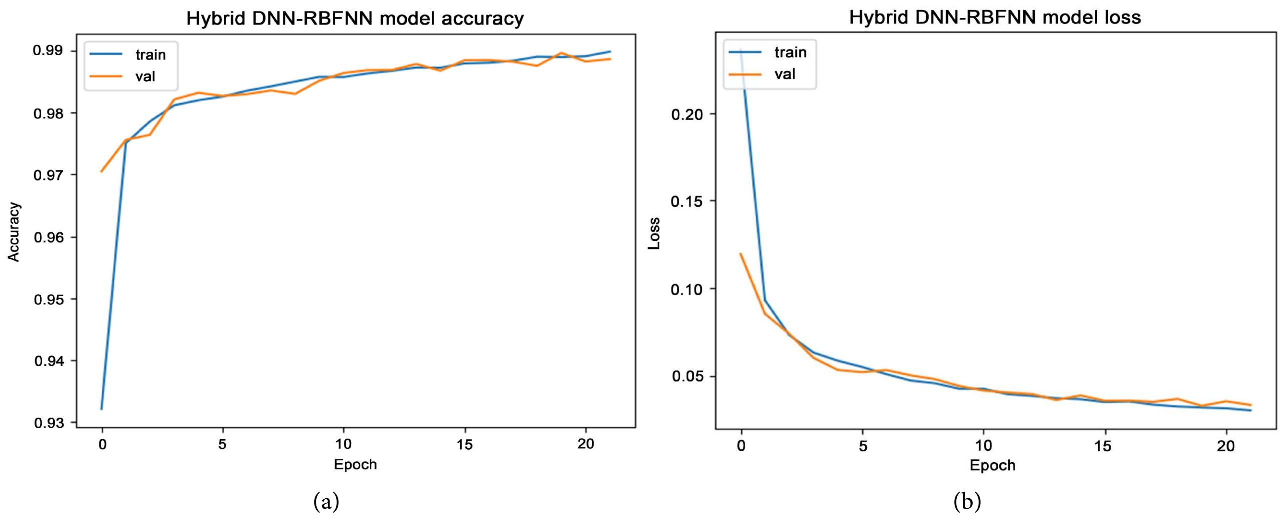


Figure 8. Accuracy and loss graphs.

4.6. Performance Metrics

Our developed model achieved highly favorable outcomes, demonstrating an impressive overall detection accuracy of 98.80%, surpassing the accuracy of 97.70% for DNN and 97.81% for RBFNN. Table 2 provides a concise overview of the sensitivity, specificity, and accuracy of our developed model compared to the Dense Neural Network (DNN) and Radial Basis Function Neural Network (RBFNN) as discussed in the literature. It is worth noting that the DNN and RBFNN exhibited inadequate performance in identifying R2L. In contrast, our developed model not only achieved better overall accuracy but also effectively addressed this limitation, providing a solution that outperformed the other models.

Table 2. Evaluation Metrics for DNN, RBFNN, and DNN-RBFNN.

Evaluation Metrics				
Model	Class	Specificity	Sensitivity	Overall Accuracy
DNN	Dos	0.994	0.982	0.977

Continued

DNN	Probe	0.993	0.965	0.977
	U2R	0.989	0.942	
	R2L	1.0	0.0	
	Normal	0.993	0.980	
RBFNN	Dos	0.993	0.982	0.978
	Probe	0.994	0.953	
	U2R	0.993	0.899	
	R2L	1.0	0.133	
	Normal	0.989	0.986	
DNN-RBFNN	Dos	0.997	0.993	0.988
	Probe	0.995	0.990	
	U2R	0.995	0.937	
	R2L	0.998	0.307	
	Normal	0.997	0.989	

4.7. Adversarial Examples Comparison

Like any other neural network model, our developed model was also susceptible to adversarial examples. Figure 9 illustrates the comparison between the developed model and the standalone models of DNN and RBFNN.

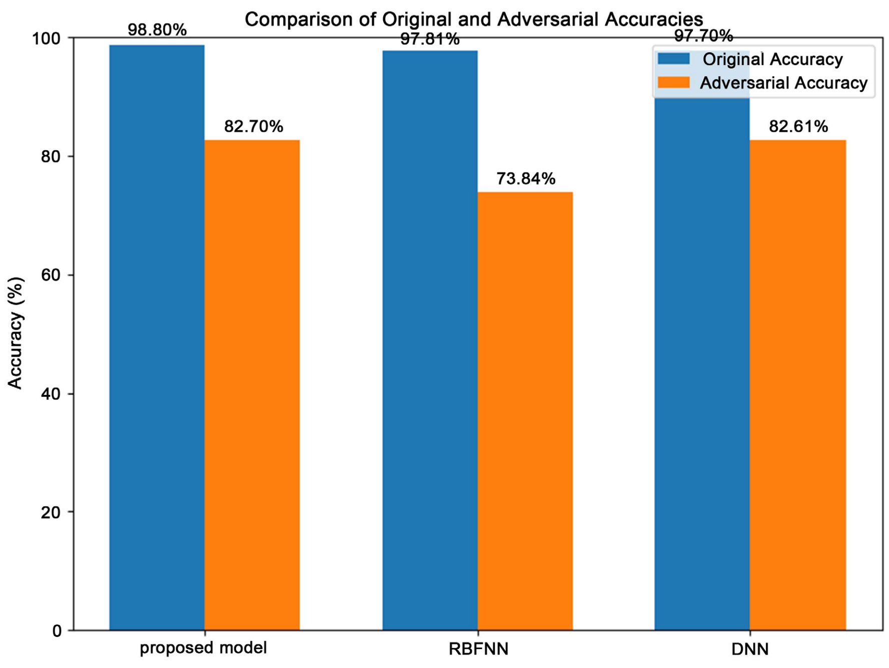


Figure 9. Adversarial examples bar plots.

The RBFNN model achieved an accuracy of 97.81% on the original dataset. However, its adversarial accuracy dropped significantly to 73.84%, suggesting vulnerability to perturbed inputs. Similarly, the DNN model achieved an accuracy of 97.70% on the original dataset, but its adversarial accuracy decreased to 82.61%, indicating susceptibility to adversarial examples. The developed model demonstrated the highest accuracy at 98.80% on the original dataset. However, like the RBFNN and DNN models, its adversarial accuracy decreased to 82.70%. Although still relatively high, this indicates that the developed model is also susceptible to adversarial attacks to some extent.

4.8. Discussion of Results

Accuracy indicates how often the classification model accurately determines whether the network packets sent could result in a normal connection, probe, R2L (Remote to local), U2L (User to root), or DoS (Denial of Service) attack. Sensitivity evaluates the ability of the classifier to make accurate positive predictions across all the different classes.

It measures how well the model can identify true positive cases, indicating its capability to detect intrusions correctly. On the other hand, specificity assesses the model's ability to make accurate negative predictions. It measures how well the model can identify true negative cases, indicating its capability to correctly classify non-intrusive in-stances. The results obtained from the Dense Neural Network were consistent with those obtained by [6]. The DNN model demonstrates good performance in terms of specificity and sensitivity for most classes, except for the R2L class where the sensitivity is 0 and an overall accuracy of 0.977.

The RBFNN model shows comparable performance to the DNN model, with high specificity and sensitivity foremost classes. However, it struggles with the R2L class, where the sensitivity is only 0.133 and an overall accuracy of 0.978. Our developed model achieved the highest specificity and sensitivity values for most classes, including a significant improvement in the R2L class compared to the other models. However, the sensitivity for R2L is still relatively low at 0.307, and the overall accuracy is 0.988.

5. Conclusion and Recommendations

Through the utilization of advanced deep learning algorithms, particularly DNN and RBFNN, the outcomes derived from comparing the performance of the hybrid DNN-RBFNN model against individual DNN and RBFNN models were nothing short of remarkable. The results showcased the remarkable capabilities of the hybrid model, surpassing the individual DNN and RBFNN models across a range of metrics. This underscores the immense potential and efficacy of synergizing these two powerful approaches to bolster network security and fortify defenses against potential threats.

Furthermore, considering the escalating complexity and extensive nature of equipment safety administration, coupled with the ever-growing security needs, the future of network security technologies may rely on the integration of diverse deep learning tools. In this context, the efficacy of DNN and RBFNN models has been extensively discussed in this study. The experimental results documented the superiority of the hybrid DNN-RBFNN model over standalone DNN and RBFNN models.

Looking forward, the future of network security technologies may involve the continued integration and advancement of various deep learning tools, including the incorporation of hybrid deep convolutional neural network models, to ensure comprehensive network security and address emerging threats.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Shanthi, S.A. and Sathiyapriya, G. (2022) Universal Approximation Theorem for a Radial Basis Function Fuzzy Neural Network. *Materials Today*: *Proceedings*, **51**, 2355-2358. https://doi.org/10.1016/j.matpr.2021.11.576

[2] Manu, B. (2016) A Survey on Secure Network: Intrusion Detection & Prevention Approaches. *American Journal of Information Systems*, **4**, 69-88.

[3] Roesch, M. (1999) Snort: Lightweight Intrusion Detection for Networks. *Lisa*, **99**, 229-238.

[4] Parul, B. and Kumar, G. and Rajiv (2014) A Review on Attacks Classification Using Decision Tree Algorithm, *Int J*, **2**.

[5] Thirimanne, S.P., Jayawardana, L., Yasakethu, L., Liyanaarachchi, P. and Hewage, C. (2022) Deep Neural Network Based Real-Time Intrusion Detection System. *SN Computer Science*, **3**, Article No. 145. https://doi.org/10.1007/s42979-022-01031-1

[6] Zarai, R., *et al.* (2020) Recurrent Neural Networks & Deep Neural Networks Based on Intrusion Detection System. *Open Access Library Journal*, **7**, 1. https://doi.org/10.4236/oalib.1106151

[7] Vinayakumar, R. and Soman, K.P. and Poornachandran, P. (2017) Applying Convolutional Neural Network for Network Intrusion Detection. 2017 *International Conference on Advances in Computing, Communications and Informatics* (*ICACCI*), Udupi, 13-16 September 2017, 1222-1228. https://doi.org/10.1109/ICACCI.2017.8126009

[8] Miyato, T. and Dai, A.M. and Goodfellow, I. (2016) Adversarial Training Methods for Semi-Supervised Text Classification. arXiv:1605.07725.

[9] Chen, Z. and Qian, P. (2017) Application of PSO-RBF Neural Network in Network Intrusion Detection. 2009 *Third International Symposium on Intelligent Information Technology Application*, Nanchang, 21-22 November 2009, 362-364. https://doi.org/10.1109/IITA.2009.154

[10] Otoum, Y. and Nayak, A. (2021) As-Ids: Anomaly and Signature-Based Ids for the Internet of Things. *Journal of Network and Systems Management*, **29**, 1-26. https://doi.org/10.1007/s10922-021-09589-6

[11] Dina, A.S. and Manivannan, D. (2021) Intrusion Detection Based on Machine Learning Techniques in Computer Networks. *Internet of Things*, **16**, Article ID: 100462. https://doi.org/10.1016/j.iot.2021.100462

[12] Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998) Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, **86**, 2278-2324. https://doi.org/10.1109/5.726791

[13] Guyon, I. and Elisseeff, A. (2003) An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, **3**, 1157-1182.

[14] Powell, A., Bates, D., Van Wyk, C. and de Abreu, D. (1998) A Cross-Comparison of Feature Selection Algorithms on Multiple Cyber Security Data-Sets. *FAIR*, **86**, 196-207.

[15] Lin, Y., Zhao, H., Tu, Y., Mao, S. and Dou, Z. (2020) Threats of Adversarial Attacks in DNN-Based Modulation Recognition. *IEEE INFOCOM*, 2020-*IEEE Conference on Computer Communications*, Toronto, 6-9 July 2020, 2469-2478. https://doi.org/10.1109/INFOCOM41043.2020.9155389

[16] Afaq, S. and Rao, S. (2020) Significance of Epochs on Training a Neural Network, *International Journal of Scientific and Technology Research*, **19**, 485-488.