STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–16.
Published online in International Academic Press (www.IAPress.org)

Explainable Deep Neural Network for a Reliable Intrusion Detection System with Shapley Additive Explanation Method

Fikri Mulyana Setiawan, Dodi Devianto*, Mawanda Almuhayar

Department of Mathematics and Data Science, Universitas Andalas, Indonesia

Abstract The escalating complexity of cyber-attacks and the severe consequences of data breaches necessitate a shift toward more advanced, yet accountable, network security infrastructures. While deep learning models offer superior performance in identifying intrusions, their inherent ”black-box” nature hinders practical adoption in critical environments where security analysts must verify alerts, justify defensive actions, and conduct forensic audits. To address this lack of transparency, this study proposes an explainable Deep Neural Network (DNN) framework for a reliable Intrusion Detection System (IDS) using the CIC-IDS2017 dataset. By integrating the Shapley Additive Explanations (SHAP) method, we bridge the gap between high-performance detection and interpretability. Our experimental results demonstrate that our minimalist DNN architecture achieves an outstanding accuracy of 99.57% and an AUC-ROC of 0.9997, maintaining high detection rates across various attack types with significantly lower computational overhead compared to complex hybrid models. Furthermore, the SHAP analysis identifies Flow IAT Std and Packet Length Variance as the most influential features, offering granular insights into the model's reasoning. This research demonstrates that high-performance deep learning can be paired with rigorous interpretability, providing a robust and transparent solution for real-time network security monitoring.

Keywords Intrusion Detection, Deep Neural Network, Explainable AI, SHAP values, CIC-IDS2017

AMS 2010 subject classifications 68T05, 68T10, 68T37

DOI: 10.19139/soic-2310-5070-3141

1. Introduction

Data security is increasingly important in the current digital era, where information has become a highly valuable asset. The widespread use of communication technology has led to a substantial increase in the volume of data transmitted through the network [1]. This high volume and rapid transmission of data through the communication network initiates several cybercrime activities, such as intrusion, which can compromise data confidentiality and integrity. The impact of intrusion extends outside technical issues, as cybercrime activities, such as intrusion, can lead to negative consequences, including financial losses, data misuse, diminished customer trust and damage to the reputation of an organization. In recent years, different methods have been adopted to identify intrusion, allowing authorities to intervene before problems become more severe.

According to the National Institute of Standards and Technology (NIST), an intrusion is a security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system or system resource without having authorization to do so [2]. In the context of cybercrime, an intrusion occurs when an intruder enters or attempts to enter a communication network system to commit various crimes, such as data theft. To address intrusion before it becomes more serious, an intrusion detection system was developed.

*Correspondence to: Dodi Devianto (Email: ddevianto@sci.unand.ac.id). Department of Mathematics and Data Science, Universitas Andalas. Jln. Kampus Limau Manis, Padang, 25163, Indonesia.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

The most critical component of an intrusion detection system is the classification engine, which determines how network data flow represents normal activity or an intrusion. Several methods are applied to classification engines for intrusion detection, including classic methods such as fuzzy logic [3], machine learning [4], and more sophisticated methods comprising neural network models [5], [6]. Several previous studies showed that neural network-based models were capable of better extracting high-dimensional data representations [7], [8], for effective use in network intrusion detection systems. This is evident in the performance of neural network-based model, which outperform various other machine learning versions in multiple studies. Training a neural network-based model does not require a separate feature selection process, different from classical machine learning methods, reducing the need for human intervention [7], [9]. This minimizes the potential for human error that may occur during model training.

Although neural network-based models excel in various aspects, there are shortcomings when implemented for intrusion detection problems. This model is often regarded as a black-box because the prediction processes are challenging to understand and interpret [10]. The issue is certainly a major obstacle in implementing neural network-based models in network intrusion detection systems, where explanations and rationale behind model prediction results are crucial information for network security analysis, as well as to increase confidence in the outcomes. To address this issue, Explainable AI (XAI) methods were introduced to develop methods that make AI model results understandable and interpretable [11].

Since the initial introduction of Explainable AI in 1980, numerous studies have been conducted to improve the interpretability of AI models and increase public trust [12]. In 2015, the Defense Advanced Research Projects Agency (DARPA) introduced the XAI program, designed to improve the development of AI systems that are both interpretable and understandable, while also increasing public trust by providing a clearer understanding of the reasoning behind model prediction results [13]. In a similar way, the European Union passed regulations on the right to algorithmic explanation, allowing individuals to obtain a clarification of how decision-making processes are conducted based on data. This change in AI studies shifted from initially focusing solely on improving model performance metrics to developing model that performed superiorly and were also interpretable and explainable.

Based on the ideas mentioned earlier, this study uses a Deep Neural Network (DNN) model incorporated with XAI methods for a network intrusion detection system. The incorporation allows the analysis to obtain the predictions of the model and also the reasons for the results, providing understanding and increasing confidence in its outcomes when handling intrusion detection cases. While various studies have explored XAI and deep learning in intrusion detection using complex architectures such as Attention-CNN-LSTM or CNN-BiLSTM, this research distinguishes itself by focusing on the efficiency-performance trade-off, operational viability and model interpretability. The main contributions of this paper are explicitly stated as follows:

- Computational Efficiency. We demonstrate that a minimalist DNN architecture with approximately 20,165 parameters can outperform the accuracy of more complex hybrid models on the CIC-IDS2017 dataset.
- Operational viability. While benchmark models like Random Forest achieve comparable evaluation metrics, they lack operational viability because of excessive prediction latency and substantial model size. It is unsuitable for deployment stage in resource-constrained devices.
- Model Interpretability. We provide a rigorous model explainability analysis, incorporating both local and global interpretation to understand the model's reasoning behind each prediction.

2. Previous Studies

There are several studies conducted to develop an effective and efficient model for intrusion detection. For example, Miguel Arcos-Argudo *et al.* [14] studied different classical machine learning models, such as Naive Bayes (NB), Logistic Regression (LR), and Linear Disriminant Analysis (LDA). This study compared two approaches for handling imbalanced data in CIC-IDS2017 dataset, with and without sampling techniques. The result demonstrate that data augmentation techniques such as SMOTE didn't affect the model performance. LR and LDA achieve a high accuracy, precision and recall, above 98% even without data augmentation techniques. This result shows that

the model capable to achieve a good performance without data augmentation techniques even though CIC-IDS2017 dataset is heavily imbalanced.

A recent study by Ali *et al.* [15] asserts that the implementation of the Synthetic Minority Oversampling Technique (SMOTE) significantly enhances the performance of both classical machine learning and deep learning models. While the authors identify the use of SMOTE as a primary contribution of their work, a critical methodological gap remains because the study does not provide a comparative analysis of model performance trained without the oversampling technique. Consequently, without a baseline performance for comparison, it is difficult to empirically validate whether the reported improvements are uniquely attributable to the SMOTE implementation or other architectural factors.

In recent years, deep learning models have gained significant traction in the field of intrusion detection due to their robust capability in extracting complex, non-linear patterns from high-dimensional network traffic data. This trend has led to the development of increasingly sophisticated architectures designed to enhance detection precision across diverse attacks. A notable example is the work of Alashjaee, who proposed a deep learning model integrated with an attention mechanism, named Attention-CNN-LSTM, evaluated on the NSL-KDD and Bot-IoT datasets [16]. While this model achieved an accuracy range of 94.8% to 97.5%, the study further highlights a critical performance metric regarding its operational efficiency. The authors reported that the architecture maintains an average latency of 32ms per sample, a figure achieved using high-end NVIDIA RTX 2080 Ti hardware, which served as the basis for their claim regarding the model's suitability for real-time deployment in high-traffic environments. However, this assertion overlooks the practical constraints of real-world infrastructure, as the heavy reliance on such high-performance GPU specifications may not be feasible for standard network middleboxes or resource-constrained edge devices. Consequently, the reported latency provides an idealized performance benchmark that may not accurately reflect the model's viability in actual deployment scenarios where high-end computational resources are unavailable.

As the demand for explainable artificial intelligence (XAI) continues to escalate, the research trend in deep learning has shifted significantly from a singular focus on detection performance toward the development of inherently interpretable and transparent models. A notable contribution to this shift was made by Younisse *et al.* [17], who attempted to provide a statistical explanation for feature importance derived from the SHAP method by employing density function analysis based on Kernel Density Estimation (KDE). While their approach offers valuable insights into the global behavior of the model by correlating statistical distributions with feature relevance, the scope of their analysis remains limited to a high-level, aggregate perspective. This focus on global interpretation overlooks a fundamental requirement of cyber threat analysis, which is the need for granular, local interpretation. In a practical security operations context, it is imperative to understand why a specific network transaction or individual packet has been flagged as malicious.

A different approach was introduced by Mahmoud *et al.* [18], who proposed an explainable two-stage detection process for multi-class classification using the CIC-IDS2017 and UNSW-NB15 datasets. This framework employs a sequential methodology where the first stage focuses on binary classification to distinguish between benign and malicious traffic, while the second stage serves as a specialized classifier to identify the specific type of each detected attack. To achieve high detection performance, the study utilizes a diverse ensemble of models across both stages, including LightGBM, XGBoost, and Bidirectional LSTM. While this multi-stage approach achieves impressive accuracy metrics across multiple datasets, the study presents two critical limitations that undermine its practical utility. First, it lacks critical analysis regarding operational viability, specifically omitting the detection latency required to validate real-time deployment scenario. Second, the explainability analysis provided in this study is constrained to global interpretation, which describes general model behavior but fails to offer the granular, local insights required for a case-specific forensic analysis of individual network threats.

To mitigate the issue of excessive detection latency, some researchers advocate for the use of classical machine learning models due to their lower computational demands. Le *et al.* [19] followed this approach by employing Explainable Decision Tree (DT) and Random Forest (RF) models on the NF-BoT-IoT-v2 and NF-ToN-IoT-v2 datasets. Although their models achieved near-perfect accuracy, the study provides no empirical justification for detection latency in real-time deployment. Furthermore, while SHAP was utilized for global and local explanations,

the analysis remains primarily descriptive of the plots. The study fails to provide a deep, domain-specific rationale for the feature contributions, leaving the actual cause of an intrusion classification semantically unexplained.

3. Theoretical Review

3.1. Neural Network

Neural networks first appeared in the early 1940s with the ability to model data using nonlinear functions [8]. What distinguishes neural networks from other machine learning models is the architecture, which consists of several layers. Each sheet in a neural network model comprises nodes, which are computational units representing the values and weights connecting nodes between layers [20], [21]. In addition, the simplest neural network architecture consists of only two types of layers, namely input and output. Neural network models with this architecture are also referred to as single-layer neural networks or perceptrons [22], as shown in Figure 1.

The calculation of node values in the output layer is based on equation (1), where n represents the number nodes in the input layer and g is the activation function.

$$\hat{y} = g(\sum_{j=1}^{n} w_j x_j + b) \tag{1}$$

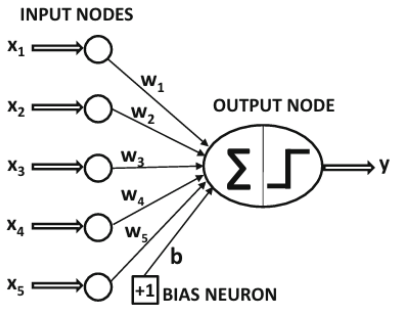

Figure 1. Illustration of Perceptron Architecture.

A more complex neural network architecture consists of three types of layers, namely input, hidden, and output [23], which is also called a Multi-Layer Perceptron (MLP). In the MLP architecture, the node values in the input layer are gradually passed to the output layer through hidden layers with appropriate weights, as shown in Figure 2.

The computational process in the MLP model is conducted in the same way as in the perceptron. The values at the input nodes are used to calculate the activation values at each node in the hidden layer, which are then forwarded to the output layer. The values at the nodes in the hidden layer, namely A_1, A_2,..., A_K, are calculated using equation (2).

$$A_k^{(1)} = g\left(w_{k,0}^{(1)} + \sum_{j=1}^{p} w_{k,j} X_j\right) \tag{2}$$

Based on the activation value of the model in the hidden layer, the model prediction result, namely $\hat{Y}$ can be calculated using equation (3).

$$\hat{Y} = g\left(\beta_0 + \sum_{k=1}^{K} w_k A_k\right) \tag{3}$$

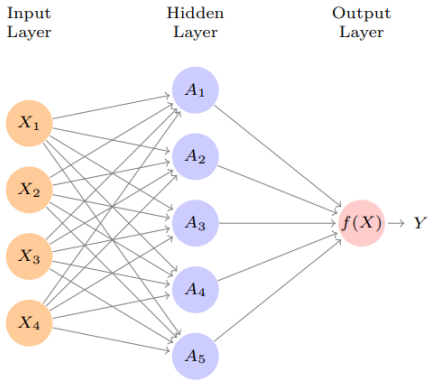

Figure 2. MLP Architecture with One Hidden Layer.

3.2. Deep Neural Network (DNN)

Modern neural network models typically consist of more than one hidden layer. Neural network models with this architecture are also referred to as DNN, as shown in Figure 3.

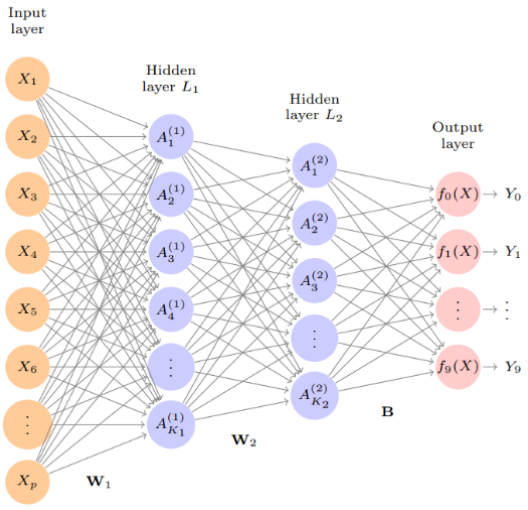

Figure 3. MLP Architecture with Multiple Hidden Layer.

DNN model performs the same computational processes as a regular neural network, but with more hidden layers. For example, when p features are present in the data with variable X and K node units in the first hidden layer, then there are $K \times (p+1)$ weights ($w_{(k,j)}^{(1)}$) connecting the input layer to the first layer. The weights $w_{(k,j)}^{(1)}$ can be represented as a matrix W_1, signifying the model weights in the first hidden layer, where:

$$\mathbf{W}_1 = \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} & \dots & w_{1,p}^{(1)} & w_{1,0}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} & \dots & w_{2,p}^{(1)} & w_{2,0}^{(1)} \\ w_{3,1}^{(1)} & w_{3,2}^{(1)} & w_{3,3}^{(1)} & \dots & w_{3,p}^{(1)} & w_{3,0}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{K,1}^{(1)} & w_{K,2}^{(1)} & w_{K,3}^{(1)} & \dots & w_{K,p}^{(1)} & w_{K,0}^{(1)} \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \\ 1 \end{bmatrix}$$

Where vector X represents the input data and W_1 signifies the model weights in the first layer. The calculation of the activation value of each node in the first hidden layer is as follows.

$$A^1 = g(\mathbf{W}_1 X) \tag{4}$$

3.3. Loss Function

A loss function represents the difference between the predicted and actual value of the model. In binary classification tasks, a commonly used loss function is the log-loss, widely known as Binary Cross-Entropy (BCE), which is expressed as follows.

$$L(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{5}$$

During training, the DNN model is optimized by minimizing the value of the loss function. This process allows optimal model parameter adjustments, leading to the smallest loss function.

3.4. Model Evaluation Metrics

Model evaluation metrics are indicators that quantitatively measure and evaluate the performance of a machine learning model. In binary classification problems, several commonly used metrics are accuracy, precision, and recall [24], [25].

1. Accuracy is a metric that shows the percentage of observations correctly classified by the model [25]. A high accuracy value signifies that the model can classify the majority of the observed data correctly. Mathematically, accuracy is calculated as,

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

   Where TP, TN, FP, and FN are True Positive, True Negative, False Positive, and False Negative, respectively.

2. Precision is a metric showing the percentage of positive classes correctly predicted by the model out of all positive classes that the model predicts. This signifies that when a model achieves 100% precision, then all positive predictions generated by the model are correct [25]

$$precision = \frac{TP}{TP + FP} \tag{7}$$

3. Recall, known as sensitivity, is a metric showing the percentage of positive classes that the model correctly predicts out of all positive classes in the dataset. Mathematically, recall is calculated as,

$$recall = \frac{TP}{TP + FN} \tag{8}$$

4. Explainable AI (XAI)

Since the 1980s, experts in the field of AI have attempted to expand the scope of AI by considering model interpretability to increase confidence in its predictions [12]. To achieve the objective, several methods have been developed to explain and interpret various machine learning models. Concerning the objective, these methods are also known as XAI methods. Several main reasons are mentioned concerning the development of XAI methods, which include the following [12], [26].

1. Improving understanding of model results.
2. Meeting industry standards. In some industries, such as healthcare, every AI system used should be explainable to ensure model transparency and accountability.
3. Increasing trust and implementation of AI. An explainable and interpretable AI model is essential for developing trustworthy as widely adopted AI systems.
4. Model Validation. Model interpretation is crucial to ensure that model results are not affected by bias and errors during the development process.

SHAP is an XAI method that aims to analyze the importance of features in a model. Feature importance analysis using SHAP is conducted by calculating the contribution of each feature in the data to the prediction results of the model [12], [27]. The SHAP method is based on a mathematical concept in game theory called the Shapley value [28]. The use of this Shapley value allows the calculation of the contribution of each feature to the results obtained by the model [27]. Suppose x is the input data and $f(x)$ is a function in the machine learning or deep learning model that provides the prediction value, then the SHAP value of the i-th – namely ϕ_i – is calculated using equation (6).

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_x(S \cup \{i\}) - f_x(S) \right] \tag{9}$$

Where F is the set of features in the input data, and S represents the subset of all features except the i-th feature. In the mentioned equation, $f_x(S)$ represents the predicted result of the model, considering only the S features in the data. Adding the i-th feature will result in the predicted result being $f_x(S \cup i)$, which shows the latest calculated outcome with the additional contribution of the i-th feature.

5. Data and Method

5.1. Data Source

This study used a dataset provided by the Canadian Institute for Cybersecurity (CIC) at https://www.unb.ca/cic/datasets/ids-2017.html. The dataset consisted of 79 columns and 2.830.743 rows representing network flows, which were the activity of sending a series of data packets between two points (nodes) on network during a single communication session. Each row in the data contained detailed information about the communication between nodes on the computer network, including Flow Duration, Packet Length, and other relevant details. Various attack types in this dataset are collectively labeled as the 'Attack' class. The distribution between benign and attack classes is shown in the Figure 4.

5.2. Data Preprocessing

Data obtained from the CIC website passed through a preprocessing stage before being used to train a DNN model. The preprocessing stages included data splitting, handling missing values, transforming categorical data,

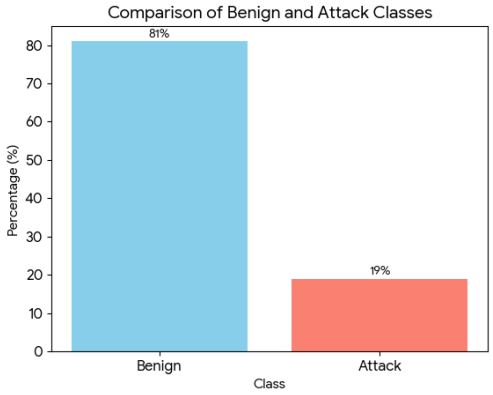

Figure 4. Distribution of Benign and Attack classes in the CIC-IDS2017 dataset.

and feature scaling. To ensure the integrity of the results, all preprocessing parameters were derived exclusively from the training set to prevent data leakage.

Data Splitting. At this stage, the data was split into training, validation, and testing sets by using stratified splitting techniques to ensure the consistency of class proportion on each set of data. Training set was used to train the model, validation set was applied to assess model performance and perform hyperparameter tuning, while the testing set was used to evaluate the final model results as shown in Table 1.

Table 1. Data Distribution across Training, Validation, and Testing Sets

	Training	**Validation**	**Testing**
Count	1,243,706 flows	266,509 flows	266,509 flows
Percentage	70%	15%	15%
Class Proportion			
Benign	81%	81%	81%
Attack	19%	19%	19%

Categorical data transformation. In the dataset, a categorical variable/feature was shown, namely ”Label,” which had a value of ”BENIGN,” representing a specific type of attack. All attack types in this dataset were then transformed into one class, ”ATTACK.” Therefore, the ”Label” feature had two possible values, namely ”BENIGN” and ”ATTACK.” The ”BENIGN” class was transformed to 0, and the ”ATTACK” class was converted to 1. During training, the DNN model only needed to determine how an activity should be categorized as 0 or 1.

Feature scaling. Feature scaling is the process of transforming numeric/quantitative values into a specific scale/interval, for example, from -1 to 1. Several commonly used methods for this transformation include min-max scaling, mean scaling, and z-score scaling (also known as standardization) [8]. In this study, the standardization method was used to transform the numerical features in the dataset, having a mean of μ and a standard deviation of σ. Transformation with standardization was conducted during the process using the following equation.

$$Z = \frac{X - \mu}{\sigma} \tag{10}$$

6. Result and Discussion

6.1. Experimental Setup

The experimental procedures were executed within a Kaggle Notebook environment powered by an Intel(R) Xeon(R) CPU @ 2.20GHz and 31 GB of RAM. To ensure research reproducibility, a random seed of 42 was consistently applied for data partitioning and weight initialization. The software stack included Python 3.12 as the base language, with TensorFlow 2.19 for deep learning implementation and Scikit-learn 1.6.2 for data preprocessing and evaluation metrics. During the deep learning model training, dual NVIDIA Tesla T4 GPUs with a combined 30 GB of VRAM were utilized to accelerate computational performance.

6.2. DNN Model Construction

After preprocessing, the data was used to train a DNN model capable of performing intrusion detection tasks. The following was the combination of architecture and hyperparameters used in this study after various tests.

1. Model Architecture. The DNN model in this study consisted of five layers, namely an input, an output, and three fully-connected layers as hidden layers, which had 32, 64, and 32 neurons, respectively. The Figure 5 illustrate the architecture of our proposed model, with the total of 20,165 parameters.

2. Batch Size. In this study, the batch size used for training for each epoch was 128.

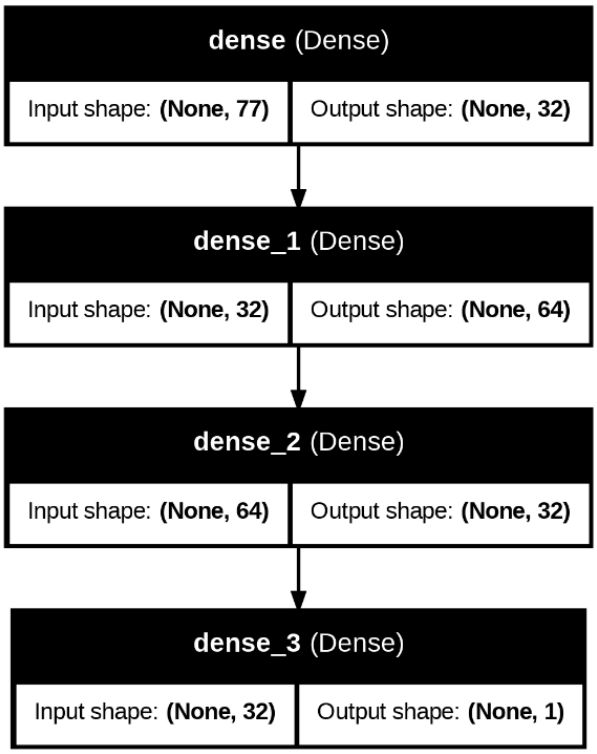

Figure 5. Architecture of the proposed Deep Neural Network (DNN)

3. Activation function. Since the DNN model in this study was developed for binary classification purposes, the activation function used was a sigmoid function on the output layer, which produced prediction probabilities. On the hidden layers, the activation function used was ReLU function.

4. Optimizer. The model was trained using the Adam optimizer with 100 epochs and a learning rate of 0.0001 with early stopping. An epoch represented the number of iterations performed during model training. This iterative training process enabled the model to learn more effectively from the data.

5. Loss function. The loss function used to evaluate the performance of this model was BCE. This function was selected because the DNN model was trained to perform binary classification.

6. Early stopping. An Early Stopping mechanism was implemented to prevent overfitting by terminating the training process once the validation loss ceased to decrease. The early stopping callback was monitored the validation loss metrics and configured with a patience of 10 epochs.

6.3. Model Evaluation

Model performance at each epoch was shown using a learning curve plot. The learning curve demonstrates accuracy achieved by the model at each epoch, which served as a baseline for model evaluation. Figure 6 illustrates the learning curve of the Deep Neural Network (DNN) model during the training and validation sets of data. The model exhibited a rapid convergence rate, characterized by a sharp increase in accuracy within the initial 20 epochs. This trend indicates that the model efficiently captured the underlying patterns of the dataset from the early stages of training. Although the maximum training limit was set to 100 epochs, the optimization process terminated at 97th epoch. This premature termination was governed by an early stopping mechanism with a patience of 10, designed to mitigate overfitting and redundant computation once the validation loss failed to decrease for 10 consecutive epochs. The marginal gap between the training and validation accuracy demonstrates the model's robust generalization capability in classifying network intrusions with high precision.

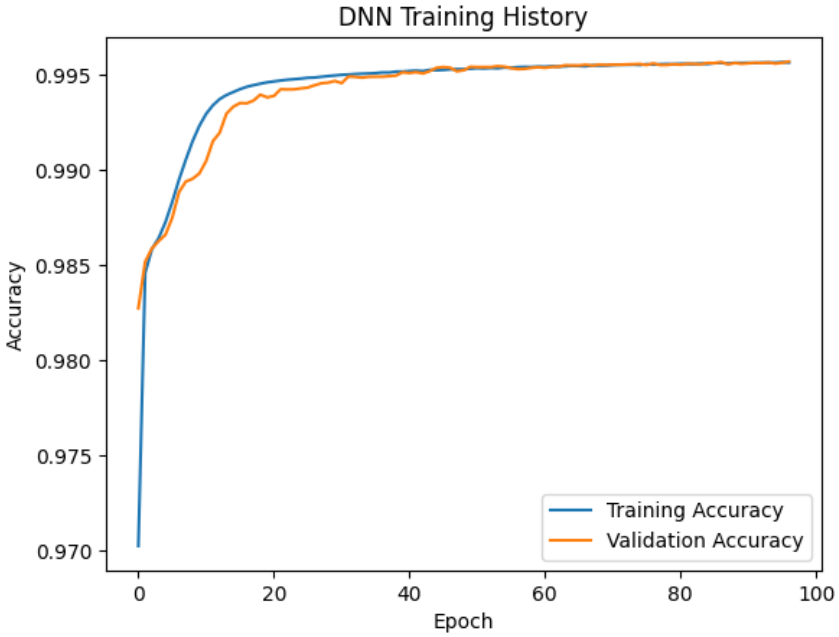

Figure 6. Learning Curve for DNN Model Training

Following 97 training epochs, the model achieved an optimal accuracy of 99.6% on the test set. The confusion matrix in Table 2 showed a comparison of the prediction results of the model with the actual classes. The table also showed that the model correctly predicted nearly all normal flows (BENIGN) and attacks (ATTACK) on the testing set. Based on Table 2, the model evaluation metrics could be calculated, as shown in Table 3.

Table 2. Confusion Matrix of Model Prediction Results on Testing Set

		Actual Value	
		ATTACK (1)	BENIGN (0)
Predicted Value	ATTACK (1)	49686 flows (98.34%)	366 flows (0.17%)
	BENIGN (0)	790 flows (1.57%)	215667 flows (99.83%)

Table 3. Classification Report for Testing Set

Class	**Precision**	**Recall**	**F1-Score**	**Support**
ATTACK	0.9927 ($\pm$ 0.000746)	0.9843 ($\pm$ 0.00108)	0.9885	50,476
BENIGN	0.9964 ($\pm$ 0.000254)	0.9983 ($\pm$ 0.00017)	0.9973	216,033
Accuracy				0.9957 ($\pm$ 0.000249)
Balanced Accuracy				0.9913
AUC-ROC Score				0.9997

The classification performance of the developed Deep Neural Network (DNN) model on the testing set is summarized in Table 3. The model achieved an exceptional overall accuracy of 99.57% ($\pm$0.0002), demonstrating its robust capability in identifying complex network traffic patterns. To ensure the reliability of these findings, a 95% confidence interval was calculated for the primary metrics, including accuracy, precision, and recall. The narrow range of these intervals (e.g., $\pm$0.0007 for precision and $\pm$0.0011 for recall) signifies high statistical stability and suggests that the model's performance is consistent and less susceptible to variance in the data distribution.

Table 3 further illustrates the precision and recall scores for both classes. The model attained a precision of 99.27% for the ATTACK class, implying that when the system issues an intrusion alert, there is a very high probability that it is a genuine threat. This reliability is critical for reducing false alarms in real-world security operations. In terms of sensitivity, the recall for the ATTACK class reached 98.43%, indicating that the model successfully intercepted the vast majority of malicious activities, with only a marginal percentage of false negatives. To provide a more balanced evaluation, the F1-score was derived as the harmonic mean of precision and recall. The model achieved an F1-score of 0.9885 for ATTACK and 0.9973 for BENIGN, confirming that the architecture maintains a superior trade-off between precision and recall despite the inherent class imbalance in the CIC-IDS2017 dataset.

The convergence of the training and validation accuracy, coupled with the high balanced accuracy of 0.9913, serves as a strong indicator that the model does not suffer from overfitting. The inclusion of the 95% confidence interval further reinforces the generalization capability of the proposed DNN model, suggesting it is a reliable tool for high-precision intrusion detection in complex network environments.

Figure 7 illustrates the Receiver Operating Characteristic (ROC) curve, which visualizes the trade-off between the True Positive Rate (Sensitivity) and the False Positive Rate (1-Specificity) across various classification thresholds. The curve exhibits a sharp vertical ascent toward the top-left corner, indicating that the proposed DNN model achieves a high detection rate while maintaining a negligible false positive rate.

Quantitatively, the model attained an Area Under the Curve (AUC) of 0.9997. This near-perfect score signifies exceptional discriminative power, implying that there is a 99.97% probability that the model will rank a randomly chosen intrusion instance higher than a randomly chosen benign instance. This characteristic confirms the model's reliability in distinguishing malicious traffic from normal activities with high confidence.

To evaluate the efficacy of the proposed DNN, a comparative analysis was conducted against prior studies utilizing the same CIC-IDS2017 dataset, as summarized in Table 4. The experimental results demonstrate that the

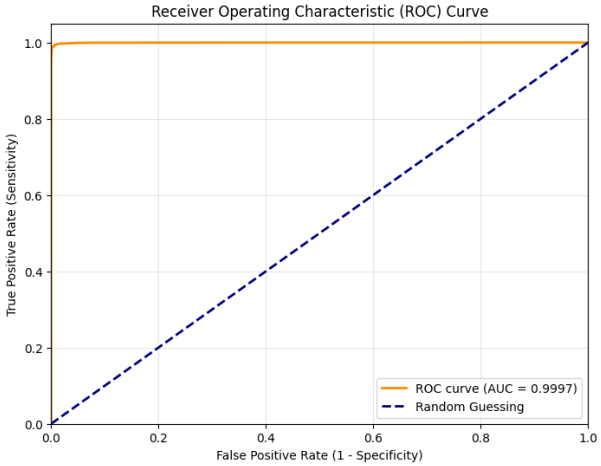

Figure 7. Receiver Operating Characteristic (ROC) Curve of the Proposed DNN

Table 4. Performance Comparison Between Proposed DNN and Prior Models on CIC-IDS2017 dataset

Model	**Accuracy** (%)	**F1-Score** (%)	**AUC** (%)
Logistic Regression [14]	98.62	97.23	99.62
BiLSTM [29]	97.72	97.75	-
CNN-LSTM [30]	93.00	81.36	-
proposed DNN	**99.57**	**98.85**	**99.97**

proposed DNN significantly outperforms existing architectures across all key performance indicators. Specifically, the proposed model achieved an accuracy of 0.9957, surpassing the BiLSTM and CNN-LSTM models, which recorded accuracies of 0.9772 and 0.9300, respectively. Furthermore, in terms of F1-Score, the proposed DNN attained a superior value of 0.9885, notably higher than the CNN-LSTM model which struggled with a significantly lower score of 0.8136. The proposed model also exhibited a near-perfect Area Under the Curve (AUC) of 0.9997, indicating a high degree of separability and robustness in classifying network traffic. These findings suggest that the optimized architecture of the proposed DNN is more adept at capturing complex patterns in intrusion detection compared to traditional recurrent and hybrid neural network approaches.

6.4. Comparative Baseline Experiments

To highlight the improvement of our model, we conduct an experiment to compare the performance of our model with 2 different machine learning classes as the benchmark, Convolutional Neural Network (CNN) as the representative of deep learning models and Random Forest (RF) with 100 estimators as the representative of classical machine learning models.

To ensure a fair comparison, the CNN was developed with an architecture analogous to the DNN. Specifically, the number of filters in each convolutional layer of the CNN was set to match the number of neurons in the corresponding hidden layers of the DNN. This structural parity was established to ensure that any differences in performance are strictly due to the model's architecture rather than a discrepancy in the number of trainable parameters.

These three models' detection performance were evaluated on a CPU-only environment to ensure a fair and unbiased comparison between deep learning and classical machine learning approaches. By utilizing a uniform hardware setup for the detection phase, the experimental results reflect the intrinsic algorithmic efficiency of each model rather than advantages gained from hardware acceleration.

Table 5. Performance Comparison of DNN, CNN, and Random Forest Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Detection Latency
DNN	99.57	99.27	98.43	98.85	4788.50 µs/flow
CNN	98.66	99.35	93.53	96.35	8979.04 µs/flow
Random Forest (RF)	99.86	99.55	99.70	99.62	26 873.20 µs/flow

Table 5 presents the performance metrics for each model. The results indicate that all models achieved comparable high detection rates, with the proposed DNN and Random Forest performing slightly better than the CNN. However, a critical divergence appears when evaluating operational viability. While the Random Forest model marginally outperforms the DNN in accuracy ($< 0.1\%$ difference), it incurs a prohibitive latency penalty. Our benchmarks reveal that the ensemble nature of Random Forest results in an detection latency of approximately 26 ms per flow, which is 6x slower than the proposed DNN ($\sim$4.8 ms). Furthermore, the proposed DNN is also superior in model size (109 KB) compared to CNN (184 KB) and RF (8.8 MB). This demonstrates that the slight accuracy advantage of traditional ensemble methods comes at the cost of significant computational overhead, making the streamlined DNN a far more viable solution for a real-time intrusion detection.

6.5. Feature Importance and Global Interpretation

SHAP, a model-agnostic method in XAI, was used in this study because of its flexibility for various neural networks and black-box models. The method was used to perform feature importance analysis, identifying which features contributed to the prediction results of the model and providing further understanding of how the model achieved its predictions. Figure 8 shows a plot of SHAP values for several data features that significantly contributed to the prediction results.

A positive SHAP value showed that the feature contributed to the prediction of the positive class. Consequently, a negative SHAP value signified that the feature contributed to the prediction of the negative class. This information brought a better understanding of how the model arrived at a particular prediction result and which features made a contribution.

Based on the plot in Figure 8, the Flow IAT Std feature emerged as the primary contributor to the model's output. Higher values of this feature strictly correlate with increased SHAP values, thereby driving the prediction toward the positive class (intrusion). From a network security perspective, this is highly intuitive. A low Flow IAT Std signifies a stable and rhythmic packet arrival interval, which is a hallmark of legitimate, non-malicious traffic. Conversely, malicious activities such as DDoS attacks often introduce bursty transmissions or irregular polling patterns. These behaviors significantly disrupt the Inter-Arrival Time consistency, leading to high standard deviation values that the DNN model correctly identifies as a robust indicator of network compromise.

The comparative SHAP analysis reveals distinct decision-making patterns across the evaluated models. The fundamental distinction between the proposed DNN model and the CNN benchmark lies in the prioritized feature space used to identify network threats. The SHAP analysis for the CNN model reveals a heavy reliance on Min Packet Length and Idle Mean, indicating that this architecture is primarily sensitive to the physical structure of individual packets and the duration of inactivity within a communication session. High SHAP values associated with low minimum packet lengths suggest that CNN tends to trigger detections based on packet-level anomalies often found in probing or scanning activities. In contrast, the DNN model shifts the focus from static packet-level characteristics to the temporal dynamics of the network flow via the Flow IAT Std feature.

When compared to the Random Forest (RF) model, the proposed DNN demonstrates a more proactive and comprehensive approach to understanding network behavior. The SHAP plot for the RF model is exclusively dominated by Backward (Bwd) features, such as Bwd Packet Length Std and Mean, reflecting the statistical properties of traffic returning from the destination to the source. This suggests that the RF model operates reactively, as its classifications are largely predicated on how a target responds to a connection, a pattern effective for identifying payload-based exploits like Web Attacks or Heartbleed. However, DNN transcends this limitation

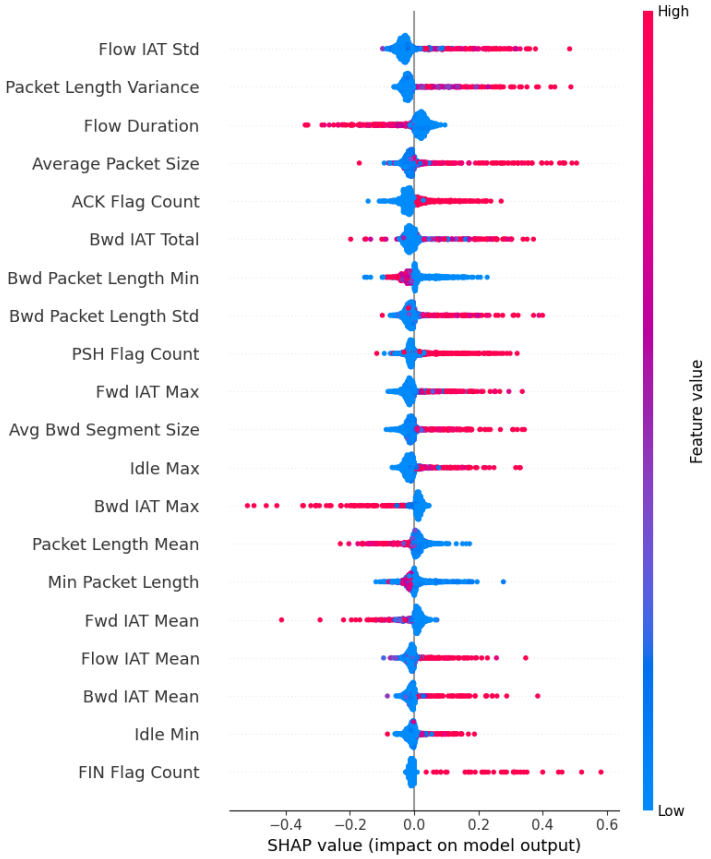

Figure 8. SHAP values of several features with the largest contribution to model prediction results

by positioning Flow IAT Std and Packet Length Variance at the top of its feature hierarchy, encompassing the characteristics of the entire bidirectional flow. By analyzing variance across the total flow duration, DNN can identify anomalies during the initial stages of an attack before a massive server response is triggered. Moreover, the significantly lower SHAP impact scale in the RF model (max 0.25) compared to DNN (max 0.6) suggests that the predictive information in RF is dispersed thinly across numerous features, whereas DNN successfully extracts more decisive and meaningful insights from complex flow dynamics.

6.6. Local Interpretation

While global analysis provides an overview of model behavior, local interpretability offers granular insights into why the DNN model classified a specific instance as an intrusion. Figure 10 shows the Waterfall plot for an instance identified by the model as an intrusion.

The transition from the baseline to the final prediction was primarily driven by several critical features. Notably, Flow IAT Std exhibited the most significant positive contribution, increasing the prediction probability by +0.33. This localized finding directly mirrors the global interpretability results, confirming that the DNN model consistently identifies high variance in packet arrival times as a primary indicator of network compromise. This is followed by Bwd Packet Length Std and Avg Bwd Segment Size, which contributed +0.19 and +0.14, respectively. The accumulation of these positive SHAP values suggests that the model identified a pattern of irregular transmission timing combined with anomalous backward traffic volume, a pattern it identified as the indicator of an intrusion.

(a) RF Benchmark Model) (b) CNN Benchmark Model

Figure 9. Global Feature Importance comparison using SHAP plots between the proposed DNN and CNN benchmark.

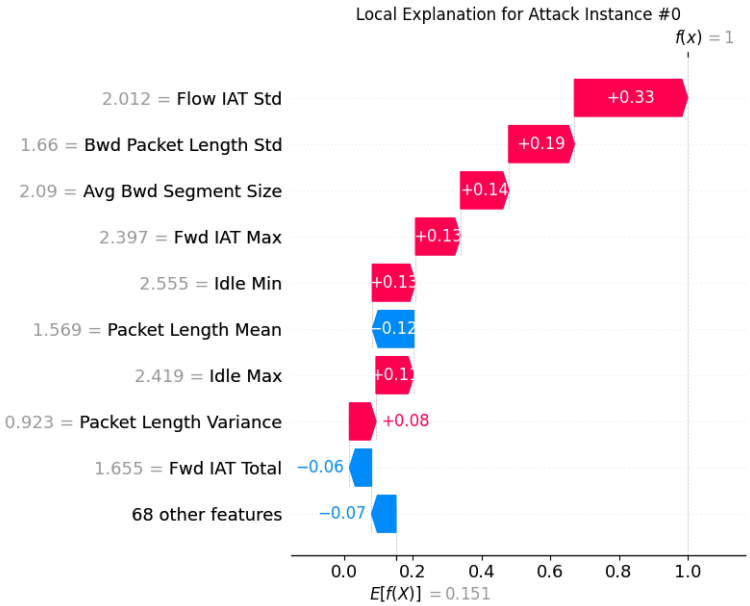

Figure 10. A decision-making process of the model when predicting an intrusion

6.7. Comparison with Previous Studies

To highlight the contribution of this research, we present a comparative analysis between our proposed approach and relevant prior studies, as summarized in Table 6.

Table 6. Comparison of the proposed model with previous studies on the CIC-IDS2017 dataset.

Study	**Method**	**Key Limitation / Gap**	**Our Improvement**
Le et al. (2022) [19]	Ensemble Trees (RF, XGBoost)	Focused on accuracy (99.9%) but ignored detection latency and model size.	Provided detailed latency analysis (∼4.8 ms vs ∼26 ms for RF) and smaller model size.
Mahmoud et al. (2025) [18]	Hybrid CNN-BiLSTM	Complex multi-stage architecture, potentially high latency. There isn't operational viability justification	Proposed a streamlined DNN for lower latency and efficient real-time deployment with comparable performance.
Alashjaee (2025) [16]	Attention-CNN-LSTM	High computational cost; latency ∼32 ms on GPU.	Demonstrated 6x faster detection speed on standard CPU with comparable performance.
Younisse et al. (2022) [17]	CNN	Focus on statistical data density (KDE) rather than security context.	Provided granular interpretability and linking SHAP values to specific attack behaviors.
Proposed Work	**Streamlined DNN**	**focus on binary classification only**	**Optimal balance of Performance, Operational Viability, and Interpretability.**

7. Conclusion

This study successfully developed a streamlined and explainable Deep Neural Network (DNN) for network intrusion detection, addressing the critical trade-off between performance, computational efficiency, and model transparency. Using the CIC-IDS2017 dataset, the proposed architecture achieved an exceptional accuracy of 99.57% ($\pm$0.0002) and a near-perfect AUC-ROC of 0.9997. These results significantly outperform more complex hybrid models, such as BiLSTM and CNN-LSTM, proving that a high-capacity yet minimalist model can capture complex non-linear patterns in network traffic more effectively.

The primary contribution of this research lies in its dual focus on operational viability and interpretability. Unlike traditional ensemble methods like Random Forest, which suffer from prohibitive detection latency, our model demonstrated a 6x faster detection speed (4.8 ms/flow) on standard CPU architecture, making it highly suitable for real-time deployment in resource-constrained environments. This minimalist architecture offers significant potential scalability, as its low parameter count ensures minimal computational overhead. This allows the system to be efficiently deployed across high-throughput network nodes to process massive traffic volumes in real-time without requiring high-end hardware.

Furthermore, the integration of the SHAP method transcended mere performance metrics by providing both global and granular local explanations. Our findings identified Flow IAT Std and Packet Length Variance as the most decisive features, shifting the detection paradigm from static packet structures to dynamic temporal behaviors of bidirectional flows. The localized waterfall plot analysis confirmed that the model consistently aligns its decision-making with established network security intuition, thereby increasing the reliability of and trust in AI-driven security operations.

While the proposed minimalist DNN achieves high performance, this study is primarily limited by its focus on binary classification. Future research will extend the architecture to multi-class scenarios for more granular threat identification and evaluate its generalization across diverse benchmark datasets, such as UNSW-NB15 or ToN_IoT.

Acknowledgement

This study was funded by Universitas Andalas according to the research contract for Final Project Recognition in the first batch with contract number 218/UN16.19/PT.01.03/PSS/2025.

REFERENCES

1. Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, *Network intrusion detection system: A systematic study of machine learning and deep learning approaches*, Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, p. e4150, 2021, doi: 10.1002/ett.4150.
2. National Institute of Standards and Technology, "Intrusion," in *Glossary*. [Online]. Available: https://csrc.nist.gov/glossary/term/intrusion.
3. J. E. Dickerson, J. Juslin, O. Koukousoula, and J. A. Dickerson, *Fuzzy intrusion detection*, Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol. 3, pp. 1506–1510, 2001, doi: 10.1109/NAFIPS.2001.943772.
4. X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, *An Adaptive Ensemble Machine Learning Model for Intrusion Detection*, IEEE Access, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.
5. 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, doi: 10.1109/ACCESS.2019.2895334.
6. W. Jo, S. Kim, C. Lee, and T. Shon, *Packet Preprocessing in CNN-Based Network Intrusion Detection System*, Electronics, vol. 9, no. 7, p. 1151, 2020, doi: 10.3390/electronics9071151.
7. I. Mrazova and M. Kukacka, *Can Deep Neural Networks Discover Meaningful Pattern Features?*, Procedia Comput. Sci., vol. 12, pp. 194–199, 2012, doi: 10.1016/j.procs.2012.09.053.
8. R. Qamar and B. Ali Zardari, *Artificial Neural Networks: An Overview*, Mesopotamian J. Comput. Sci., pp. 130–139, 2023, doi: 10.58496/MJCSC/2023/015.
9. N. Chouhan, A. Khan, and H.-R. Khan, *Network anomaly detection using channel boosted and residual learning based deep convolutional neural network*, Appl. Soft Comput., vol. 83, p. 105612, 2019, doi: 10.1016/j.asoc.2019.105612.
10. F. Sohil, M. U. Sohali, and J. Shabbir, *An introduction to statistical learning with applications in R: by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013*, vol. 6, 2022.
11. W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Muller, *Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications*, Proc. IEEE, vol. 109, no. 3, pp. 247–278, 2021, doi: 10.1109/JPROC.2021.3060483.
12. V. Hassija et al., *Interpreting Black-Box Model: A Review on Explainable Artificial Intelligence*, Cogn. Comput., vol. 16, no. 1, pp. 45–74, 2024, doi: 10.1007/s12559-023-10179-8.
13. D. Muhammad and M. Bendechache, *Unveiling the black box: A systematic review of Explainable Artificial Intelligence in medical image analysis*, Comput. Struct. Biotechnol. J., vol. 24, pp. 542–560, 2024, doi: 10.1016/j.csbj.2024.08.005.
14. M. Arcos-Argudo, R. Bojorque, and A. Torres, *A Deterministic Comparison of Classical Machine Learning and Hybrid Deep Representation Models for Intrusion Detection on NSL-KDD and CIC-IDS2017*, Sensors, vol. 18, no. 12, p. 749, 2024.
15. M. L. Ali, K. Thakur, S. Schmeelk, J. Debello, and D. Dragos, *Deep Learning vs. Machine Learning for Intrusion Detection in Computer Networks: A Comparative Study*, Electronics, vol. 15, no. 4, p. 1903, 2024.
16. A. M. Alashjaee, *Deep learning for network security: An Attention-CNN-LSTM model for accurate intrusion detection*, IEEE Access, vol. 15, no. 1, p. 21856, 2025.
17. R. Younisse, A. Ahmad, and Q. Abu Al-Haija, *Explaining Intrusion Detection-Based Convolutional Neural Networks Using Shapley Additive Explanations (SHAP)*, Big Data Cogn. Comput., vol. 6, no. 4, p. 126, 2022, doi: 10.3390/bdcc6040126.
18. M. M. Mahmoud, Y. O. Youssef, and A. A. Abdel-Hamid, *XI2S-IDS: An Explainable Intelligent 2-Stage Intrusion Detection System*, Cybersecurity, vol. 17, no. 1, p. 25, 2025.
19. T.-T.-H. Le, H. Kim, H. Kang, and H. Kim, *Classification and Explanation for Intrusion Detection System Based on Ensemble Trees and SHAP Method*, Sensors, vol. 22, no. 3, p. 1154, 2022, doi: 10.3390/s22031154.
20. D. Devianto, P. Permathasari, M. Yollanda, and A. Wirahadi Ahmad, *The Model of Artificial Neural Network and Nonparametric MARS Regression for Indonesian Composite Index*, IOP Conf. Ser. Mater. Sci. Eng., vol. 846, no. 1, p. 012007, 2020, doi: 10.1088/1757-899X/846/1/012007.
21. M. Yollanda and D. Devianto, *Hybrid Model of Seasonal ARIMA-ANN to Forecast Tourist Arrivals through Minangkabau International Airport*, Proceedings of the 1st International Conference on Statistics and Analytics, ICSA 2019, 2020, doi: 10.4108/eai.2-8-2019.2290473.
22. C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, Springer International Publishing, 2018, doi: 10.1007/978-3-319-94463-0.
23. D. Devianto, F. Wulandari, F. Yanuar, I. Rahmi, and M. Yollanda, *An Improvement of Parameter Estimation Accuracy of Structural Equation Modeling using Hybridization of Artificial Neural Network in the Entrepreneurship Structural Model*, Appl. Math. Nonlinear Sci., vol. 8, no. 2, pp. 2279–2302, 2023, doi: 10.2478/amns.2023.1.00411.
24. J. Grus, *Data Science from Scratch: First Principles with Python*, O'Reilly, first edition ed., 2015.
25. C. Miller, T. Portlock, D. M. Nyaga, and J. M. O'Sullivan, *A review of model evaluation metrics for machine learning in genetics and genomics*, Front. Bioinforma., vol. 4, p. 1457619, 2024, doi: 10.3389/fbinf.2024.1457619.
26. S. Lundberg and S.-I. Lee, *A Unified Approach to Interpreting Model Predictions*, 2017, doi: 10.48550/ARXIV.1705.07874.
27. B. H. M. Van Der Velden, M. H. A. Janse, M. A. A. Ragusi, C. E. Loo, and K. G. A. Gilhuijs, *Volumetric breast density estimation on MRI using explainable deep learning regression*, Sci. Rep., vol. 10, no. 1, p. 18095, 2020, doi: 10.1038/s41598-020-75167-6.
28. J. Padarian, A. B. McBratney, and B. Minasny, *Game theory interpretation of digital soil mapping convolutional neural networks*, SOIL, vol. 6, no. 2, pp. 389–397, 2020, doi: 10.5194/soil-6-389-2020.
29. O. M. A. Alsyaibani, E. Utami, and A. D. Hartanto, *An Intrusion Detection System Model Based on Bidirectional LSTM*, 2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS), pp. 1–6, 2021, doi: 10.1109/ICORIS52787.2021.9649477.
30. A. Kim, M. Park, and D. H. Lee, *AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection*, IEEE Access, vol. 8, pp. 70245–70261, 2020, doi: 10.1109/ACCESS.2020.2986882.