Journal of Technology and Computer (JOTECHCOM)
Vol. 3, No. 1, February 2026, pp. 9~15
ISSN: 3048-0477 (Media Online)

Implementation of The Backpropagation Algorithm to Improve the Effectiveness of Artificial Neural Network Models in Classifying Flooding Attacks

Brata Wijaya[1], Ilham Faisal[2]
[1,2]Computer Science, Faculty of Engineering and Computer Science, Harapan University, Medan, Indonesia
[1]bratawijya12@gmail.com, [2]ilham.tiftkunhar@gmail.com

Article Info

Article history:

Received November 21, 2025
Revised February 02, 2026
Accepted February 09, 2026

Keywords:

Artificial neural network
Backpropagation
Classification
Flooding
UDP *flood*
SYN *flood*
ICMP *flood*

ABSTRACT

Flooding attacks such as UDP flood, SYN flood, and ICMP flood can disrupt network stability, requiring an effective early detection system. This study aims to build a classification model using artificial neural networks (ANN) with the backpropagation method to distinguish between normal traffic and flooding attacks. Data was collected through simulation in VirtualBox with Kali Linux as the attacker and Windows 10 as the target, and captured using Wireshark. The results of training and testing both libraries showed differences in performance between the two libraries. The PyTorch model produced a prediction accuracy of 94% for normal networks and SYN floods, and 100% for UDP floods and ICMP floods, with a total accuracy of 97%. In contrast, the TensorFlow model achieved an accuracy of 77% for normal networks, 80% for UDP floods, 95% for SYN floods, and 100% for ICMP floods, with a total accuracy of 88%. The comparison of the two models shows that a simple Multi Layer Perceptron neural network with the backpropagation method using the PyTorch library is quite effective in classifying flooding attacks.

Corresponding Author:

Brata Wijaya
Universitas Harapan Medan
Email: bratawijya12@gmail.com

1. INTRODUCTION

Today's activities are inseparable from digital devices connected to the internet, making the internet the backbone of almost all communication and data transfer activities. As dependence on digital devices increases, attacks on the internet are becoming more frequent. One of the threats that disrupts the internet is a flooding attack. Flooding attacks aim to overwhelm systems using legitimate network packets, causing systems to crash and services to be disrupted or shut down. These attacks not only disrupt system operations, but can also threaten the integrity and availability of data within the network itself [1].

Flooding attacks continue to escalate in scale and frequency, placing significant pressure on modern network infrastructures and highlighting the need for accurate and adaptive traffic-classification methods. Neural-network approaches have consistently demonstrated strong capability in capturing subtle variations within network-traffic patterns. A Deep Neural Network evaluated in an SDN environment reported an accuracy of 96.31% for distinguishing UDP and SYN flooding behavior, showing that learned representations can effectively separate high-volume anomalous traffic from legitimate flows [2]. A backpropagation-based neural architecture reached a Mean Square Error of 0.0585 when discriminating between normal traffic and Distributed Denial of Service patterns, indicating that characteristic fluctuations in packet flow can be encoded

with high precision [3]. Further evidence is provided by models such as GRU, CNN-LSTM, and label-propagation, which revealed that temporal sequences of TCP flag-timing patterns serve as reliable indicators of SYN flood activity, achieving accuracies of 93% and 96% [4]. Taken together, these findings underline the relevance of neural architectures in identifying flooding-based anomalies and demonstrate that distinct statistical and temporal signatures within network traffic can be learned effectively without depending on specific IP or port information.

Based on various studies that have been conducted previously, artificial neural networks have been proven to be capable of recognizing flooding attacks with excellent accuracy. The methods used also vary, ranging from observing incoming data traffic patterns to paying attention to signal timing in the network. This indicates that machine learning technology, particularly artificial neural networks, has great potential to distinguish between various types of flooding attacks such as UDP flood, SYN flood, and ICMP flood. With this capability, neural networks can help detect attacks more quickly and accurately than traditional methods.

This study aims to create a model that can classify three types of flooding attacks in a network, namely UDP flood, SYN flood, and ICMP flood. This model will focus on how to recognize and distinguish the unique characteristics of each type of attack. The approach used in this research is an artificial neural network, which is capable of learning from network traffic data. However, this research does not directly discuss how to overcome or protect networks from these attacks, but rather how to recognize and classify them accurately based on existing patterns.

2. METHOD

The research began by determining the objectives and scope, then continued with the collection of attack data through simulations in a virtual environment. Once the data was collected, the entire dataset was cleaned and processed so that it could be used as model input. The next stage was to design the artificial neural network architecture, set the training parameters, and run the training process using the backpropagation algorithm. After the model was trained, the next step was testing and evaluation to see if the model was able to distinguish normal traffic from the three types of flooding attacks. A complete explanation of each stage is presented chronologically in the following section on methods.

2.1 Research Framework

This study uses the CRISP-DM framework as its main workflow. This framework consists of six stages commonly used in data mining projects [5]. In this study, the process was only carried out up to the evaluation stage because the model had not yet been applied to a real network environment. CRISP-DM was chosen because this framework provides a clear structure for understanding problems, preparing data, building models, and conducting evaluations systematically.

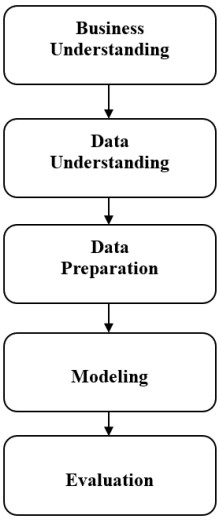

Figure 1. CRISP DM Research Framework

2.1.1 Business Understanding

The focus of the research is to build an artificial neural network model to distinguish normal traffic from three types of flooding attacks: UDP flood, SYN flood, and ICMP flood.

2.1.2 Data Understanding

Data was collected from simulations of two virtual machines in VirtualBox:

1. Kali Linux as an attack machine.
2. Windows 10 as the target.

Packets were captured using Wireshark, while attacks were sent with Hping3, which is designed as a TCP/UDP/ICMP packet delivery tool for network testing. The virtual environment was set up through static IP configuration and an internal network in VirtualBox.

2.1.3 Data Preparation

The collected data is then cleaned and processed. This process includes:

1. Converting text columns such as protocol and tcp.flags.syn into numbers.
2. Filling in empty values with 0.
3. Normalizing all features with Min–Max scaling.
4. Adding a label column (0 = normal, 1 = UDP flood, 2 = SYN flood, 3 = ICMP flood)

A sample table of cleaning and normalization results is presented in Table 3.1.

2.1.4 Modeling

The artificial neural network architecture consists of:

1. Input layer: 6 neurons (ip.src, port, protocol, frame.len, udp.length, tcp.flags.syn).
2. Hidden layer: 6 neurons, ReLU activation.
3. Output layer: 4 neurons, Softmax activation for multi-class classification.

The training method uses backpropagation with Stochastic Gradient Descent (SGD).

2.1.5 Evaluation

A confusion matrix is a method of evaluating classification models by comparing predicted results with actual labels. The results are presented in four categories: true positive (TP), which is a correct positive prediction; true negative (TN), which is a correct negative prediction; false positive (FP), which is an incorrect positive prediction; and false negative (FN), which is an incorrect negative prediction [6]. The model was evaluated using accuracy metrics and a confusion matrix to assess the system's ability to distinguish between normal traffic and three types of flooding attacks.

2.2 Research Procedures

A flowchart is a diagram that shows the sequence of processes using standard symbols so that the flow is easier to understand [7]. Flowcharts help to clearly show the logic of a program. Flowcharts are also useful for planning and documentation, as well as helping to simplify problems and improve workflows by removing unnecessary steps [8].The model training process is visualized in the flowchart in Figure 2.2 and explained in detail in the text.

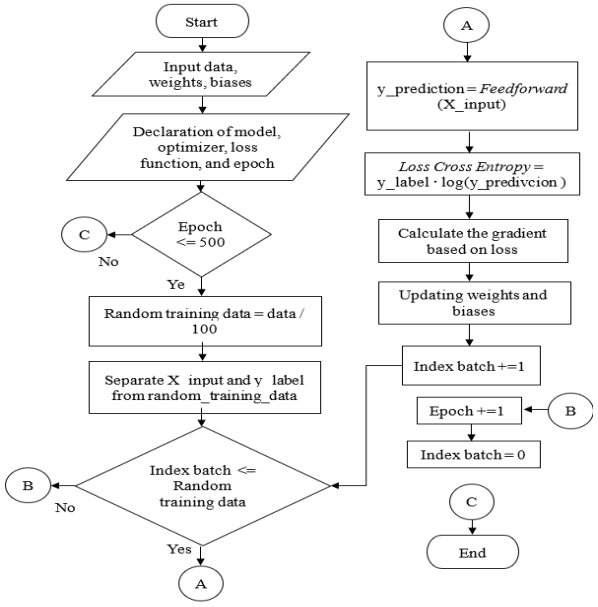

Figure 2. CRISP DM Research Framework

Pseudocode serves as a concise form of algorithm logic before being translated into code. Research shows that pseudocode can be a strong basis for generating code automatically [9]. Another study also proves that pseudocode structures help the code formation process to be more accurate [10]. In this study, pseudocode is used to summarize the model training flow so that it is easy to understand and implement. Here is the pseudocode version of flowchart for easier reading:

```
Start

input_data, weights, biases = load_input()

declare model
declare optimizer
declare loss_function
epoch = 0
batch_index = 0

while epoch <= 500:

   random_training_data = select_random_data(input_data, size = data/100)

   X_input, y_label = split_features_and_labels(random_training_data)

   while batch_index < total_batches(random_training_data):

      y_prediction = feedforward(X_input[batch_index])

      loss = cross_entropy(y_label[batch_index], y_prediction)

      gradient = compute_gradient(loss)

      weights, biases = update_parameters(weights, biases, gradient)

      batch_index = batch_index + 1

   end while

   epoch = epoch + 1
   batch_index = 0

end while

End
```

3. RESULTS AND DISCUSSION

This section presents the results of implementing artificial neural networks for classifying normal traffic, UDP floods, SYN floods, and ICMP floods. The entire process, from data cleaning and labeling to training and evaluation, is discussed systematically. The research results are presented in graphs, tables, and descriptions so that data patterns and model performance can be easily understood.

3.1. Data Preparation Results

The data from Wireshark was cleaned by converting text columns into numbers, filling in missing values, and normalizing all features using Min–Max Scaling. Each row was then labeled with a class. This process produced a dataset ready for model training.

Table 1. Representative Data Table After Data Cleaning and Normalization

ip.src	port	protocol	frame.len	udp.length	tcp.flags.syn	Label
0.8504	0.7599	0.3125	0.0861	0	0	0
0.0433	0.0008	1	0.0042	0.0334	0	1
0.0197	0.0012	0.3125	0.0042	0	1	2
0.0039	0	0	0.0042	0	0	3

3.2. Model Development Results

The model was constructed with one hidden layer containing six neurons and ReLU activation. The output layer used Softmax for four-class classification. Training was conducted for 500 epochs using backpropagation and SGD optimizer.

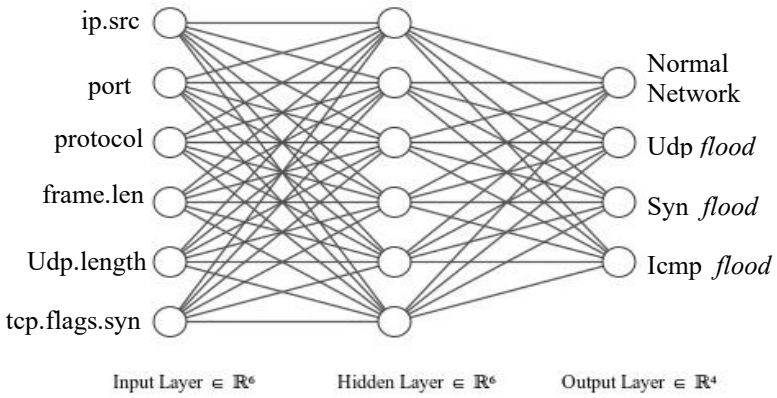

Figure 3. Neural Network Classification Model Architecture

3.3 Training Results

During training, the loss value decreased consistently. The model achieved prediction stability across all classes, as indicated by Softmax outputs that approached the true probability for each label. This is the result of training up to 500 epochs :

Epoch 10/500, Loss: 1.3921
Epoch 10/500, Loss: 1.3747
Epoch 10/500, Loss: 1.3987
Epoch 10/500, Loss: 1.3585
Epoch 10/500, Loss: 1.3777
Epoch 10/500, Loss: 1.3773
Epoch 10/500, Loss: 1.3401
Epoch 10/500, Loss: 1.4046
Epoch 10/500, Loss: 1.3814

Epoch 500/500, Loss: 0.1067
Epoch 500/500, Loss: 0.1114
Epoch 500/500, Loss: 0.0821
Epoch 500/500, Loss: 0.1055
Epoch 500/500, Loss: 0.1239
Epoch 500/500, Loss: 0.1517
Epoch 500/500, Loss: 0.0769
Epoch 500/500, Loss: 0.0536
Epoch 500/500, Loss: 0.0850

3.4 Evaluation Results

The model was tested using 100 new data points. These new data points were created to have the same pattern but with different values, producing a confusion matrix that can be seen in Table 3.2.

Table 2. Confusion Matrix

		Predict			
		Normal network (0)	UDP flood (1)	SYN flood (2)	ICMP flood (3)
Actual	Normal network (0)	13		6	
	UDP flood (1)		26		
	SYN flood (2)			26	
	ICMP flood (3)				29

From Table 3.2, we can calculate the model accuracy, which reaches 97%, which is the total accuracy of each class. The accuracy of each class reaches:

Normal Network : 94 %
Udp Flood : 100 %
Syn Flood : 94 %
Icmp Flood : 100 %

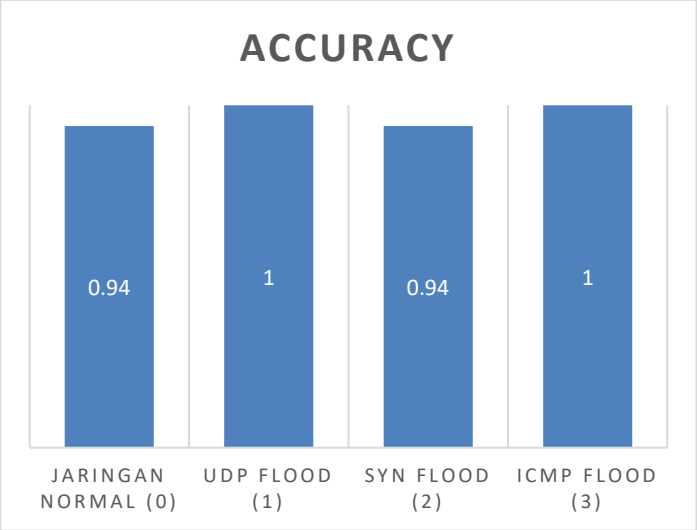

Figure 4. Accuracy of each class

3.5 Discussion

The model is able to distinguish between the four classes well, with a total accuracy of 97%. Of the 19 normal network data points, 13 were correctly identified, while all UDP flood, SYN flood, and ICMP flood data points were predicted without error. This difference in results is directly related to the variation in patterns for each class. Normal traffic has more diverse characteristics, such as TCP or TLSv1.2 protocols, variable destination ports, non-uniform packet lengths, and inactive SYN flags, making it more difficult to recognize consistently. In contrast, the three types of attacks have much more consistent patterns—UDP flood sends UDP packets to port 53 with uniform packet lengths, SYN flood shows TCP packets with active SYN flags to port 80, and ICMP flood displays ICMP packets of the same size. All three also use random source IPs, indicating spoofing. It is this consistency in patterns that makes it easier for the model to recognize attacks compared to normal traffic.

4. CONCLUSION

This study proves that artificial neural networks with backpropagation algorithms can be used to classify normal network traffic, UDP floods, SYN floods, and ICMP floods with excellent performance. The model constructed is simple, yet capable of effectively learning packet patterns captured by Wireshark after undergoing cleaning, normalization, and feature labeling processes.

The evaluation results show a total accuracy of 97%, with details of 94% for normal traffic and SYN floods, and 100% for UDP floods and ICMP floods. High performance in the attack class was influenced by

consistent and easily recognizable packet patterns, while variations in normal traffic made prediction slightly more challenging. Overall, this study shows that a simple JST approach is powerful enough to detect flooding attacks in a simulated environment, although further testing in real network conditions is still needed to ensure model generalization.

REFERENCES

[1] M. B. N. Fa'izi, "Bahaya Serangan SYN Flood dan Cara Mengatasinya," 06 November 2024. [Online]. Available: https://cyberhub.id/pengetahuan-dasar/bahaya-serangan-syn-flood.

[2] V. M. B. K. S. Mohan, "Deep Neural Network Model for UDP SYN Flood Distributed DoS Attack Detection in SDN," in *Biologically Inspired Techniques in Many Criteria Decision-Making*, Cham, 2025.

[3] A. S. G. B. S. Ahmad Fajri Khumara, "Analisa Pendeteksian Serangan DDoS Menggunakan Teori Pendekatan Neural Network Backpropagation," *CESS (Journal of Computing Engineering, System and Science),* 2022.

[4] S. Anande, Neural Network Models for TCP - SYN Flood Denial of Service Attack Detection With Source and Destination Anonymization, Rochester: Rochester Institute of Technology, 2021.

[5] J. C. ,. R. K. T. K. Pete Chapman, CRISP-DM 1.0: Step-by-Step Data Mining Guide, Chicago: SPSS Inc., 2000.

[6] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation.," *Journal of Machine Learning Technologies,* p. 37, 2011.

[7] R. W. Sebesta, Programming Language Concepts. Pearson Education, Boston, MA: Pearson Education, 2012.

[8] I. Sommerville, Software Engineering, Boston, MA: Addison-Wesley, 2011.

[9] P. P. K. C. M. L. O. P. A. A. P. L. Sumith Kulal, "SPoC: Search-based Pseudocode to Code," *NeurIPS,* 2019.

[10] M. S. D. K. Ruiqi Zhong, "Semantic Scaffolds for Pseudocode-to-Code Generation," *Association for Computational Linguistics,* 2020.