Network Intrusion Detection Using Artificial Neural Networks

ARNAB DAS*, BRAC University, Bangladesh
AVIZIT SARKAR, BRAC University, Bangladesh

This project aims to develop a network intrusion detection system (NIDS) using artificial neural networks (ANNs). The system will be trained on a dataset of network traffic patterns to identify and classify potential intrusions or attacks. The goal is to improve the accuracy and efficiency of intrusion detection compared to traditional rule-based methods. ANNs offer advantages in their ability to learn complex patterns and adapt to new threats. However, challenges such as dataset quality, model complexity, and real-time performance will need to be addressed. Ultimately, this project has the potential to significantly enhance network security by providing a more intelligent and proactive defense mechanism against cyberattacks.

ACM Reference Format:
ARNAB DAS and AVIZIT SARKAR. 2024. Network Intrusion Detection Using Artificial Neural Networks. 1, 1 (May 2024), 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Network intrusion detection is a critical component of modern cybersecurity systems, as it helps identify and mitigate potential threats and attacks on computer networks. Traditional rule-based intrusion detection systems rely on predefined rules and signatures, which can be ineffective against new or evolving attack methods. Artificial neural networks, a branch of machine learning, have shown promise in improving the accuracy and efficiency of intrusion detection by learning complex patterns and features from large datasets. Their adaptive nature allows them to recognize previously unseen attack variations. However, the success of such systems depends heavily on the quality and diversity of the training data, and they may require significant computational resources for real-time operation.

This project explores the development of an intrusion detection system using an artificial neural network trained on the KDD Cup 1999 dataset. The KDD Cup 1999 dataset is a widely used benchmark for evaluating intrusion detection systems and contains a diverse set of network connection records, including both normal traffic and various types of intrusions. While a valuable resource, the dataset has known limitations such as unrealistic traffic distributions. Therefore, this project will also investigate feature engineering techniques to improve the dataset's relevance and the neural network's performance. Additionally, the research may explore different neural network architectures to optimize accuracy and detection speed within a modern network environment.

The main objectives of this project are to:

- Preprocess the KDD Cup 1999 dataset to prepare it for training the neural network.
- Design and implement an artificial neural network architecture suitable for intrusion detection.

*Both authors contributed equally to this research.

Authors' Contact Information: ARNAB DAS, arnab.das@g.bracu.ac.bd, BRAC University, Dhaka, Bangladesh; AVIZIT SARKAR, avizit.sarkar@g.bracu.ac.bd, BRAC University, Dhaka, Bangladesh.

Manuscript submitted to ACM

- Conduct experiments to optimize the neural network's performance, including tuning hyperparameters and evaluating different data splitting strategies.
- Evaluate the trained model's accuracy and efficiency in detecting intrusions.
- Identify potential areas for further improvement and future work.

.

2 DATASET OVERVIEW

The KDD Cup 1999 dataset, specifically a 10% subset, is used in this project. This subset contains 494,021 records, each representing a network connection with 41 features and a label indicating whether the connection is normal or an intrusion.

The dataset includes a total of 23 output possibilities, with one representing a normal ("good") connection and the remaining 22 representing different types of intrusions. The intrusion types can be broadly categorized into four main groups:

- Denial of Service (DoS) attacks: Attempts to disrupt or deny legitimate users access to a system or network resources.
- User to Root (U2R) attacks: Unauthorized attempts to gain root or superuser privileges on a system.
- Remote to Local (R2L) attacks: Unauthorized access from a remote machine to gain local access on a target system.
- Probing attacks: Attempts to gather information about a network or system for potential future attacks.

Understanding the distribution and characteristics of the different intrusion types is crucial for developing an effective intrusion detection system.

3 METHODOLOGY

3.1 Data Preprocessing

Before training the neural network, the KDD Cup 1999 dataset underwent several preprocessing steps:

- Encoding: Non-numerical feature values and output possibilities were encoded with numerical values to facilitate processing by the neural network. Categorical features were encoded using one-hot encoding or label encoding techniques, while continuous features were normalized or standardized.
- Dimensionality Reduction: Principal Component Analysis (PCA) was applied to reduce the dimensionality of the data while preserving most of the relevant information. This step helps improve the training efficiency and reduces the risk of overfitting.
- Feature Scaling: MinMax scaling was used to normalize the input features, ensuring that all features contribute equally to the learning process and improving the model's accuracy.

These preprocessing steps are crucial for preparing the data in a format suitable for training the neural network and ensuring optimal performance.

3.2 Neural Network Architecture

The artificial neural network architecture used in this project consists of five layers:

one input layer, one output layer, and three hidden layers. This configuration was chosen based on experimental results, balancing accuracy and time efficiency. While a deeper network might offer potential accuracy improvements, it might also lead to overfitting and longer training times.

The input layer receives the preprocessed feature vectors from the KDD Cup 1999 dataset, while the output layer produces a binary classification indicating whether the connection is normal or an intrusion.

The three hidden layers are responsible for learning the complex patterns and relationships within the data. The model uses the Rectified Linear Unit (ReLU) activation function for the hidden layers, which introduces non-linearity and allows the network to learn complex mappings. The output layer uses the Sigmoid activation function, which is suitable for binary classification problems.

The binary cross-entropy loss function is used to measure the performance of the classification, and the Adam optimizer is employed to update the network's weights during training, as it has shown superior performance compared to other optimizers during experiments.

3.3 Hyperparameter Tuning and Optimization

To ensure optimal performance and efficiency, several experiments were conducted to tune the hyperparameters of the neural network:

Table 1. Split Ratio Accuracy

Split Ratio (training/test)	Test Accuracy
0.75 / 0.25	0.9922
0.65 / 0.35	0.9937
0.55 / 0.45	0.9932
0.45 / 0.55	0.9917
0.35 / 0.65	0.9909
0.25 / 0.75	0.9903
0.15 / 0.85	0.9887

Table 2. Number of Components Analysis

Number of components	Highest Accuracy
3	0.9937
6	0.9938
10	0.9938
12	0.9941
16	0.9945
32 (optimal)	0.9991 (optimal)
38	0.9990

- Training/Test Set Split: Different split ratios for the training and test sets were evaluated, with the 0.65/0.35 ratio yielding the highest test accuracy of 0.9937.
- Batch Size: Batch sizes of 32, 64, 128, and 256 were tested, with a batch size of 128 providing the best balance between training time and accuracy.
- PCA Components: The number of PCA components was varied to find the optimal dimensionality reduction while preserving the most relevant information. Using 32 components achieved the highest test accuracy of 0.9991.
- Early Stopping: An early stopping monitor was implemented to terminate training when no improvement in validation error was observed for a specified number of epochs, further reducing training time and preventing overfitting.

These experiments helped fine-tune the neural network's hyperparameters, ensuring optimal performance and efficiency in detecting intrusions on the KDD Cup 1999 dataset.

4 EXPERIMENTS AND RESULTS

After conducting the experiments and tuning the hyperparameters, the final model achieved a test accuracy of 0.9991 and a test loss of 0.0031, demonstrating excellent performance in detecting intrusions on the KDD Cup 1999 dataset. To evaluate the model's performance more comprehensively, various evaluation metrics were calculated, including precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics provide insights into the model's ability to correctly identify intrusions (true positives) while minimizing false positives and false negatives.

Additionally, the model's performance was analyzed across different intrusion types to identify potential strengths and weaknesses. This analysis can help guide further improvements and adjustments to the model. The training time for the final model was also recorded and compared to other configurations, demonstrating the effectiveness of the hyperparameter tuning and optimization techniques in improving the overall efficiency of the intrusion detection system.

5 FUTURE WORK

While the results achieved in this project are promising, there are several areas for potential improvement and future work:

- Exploring different encoding methods: alternative encoding techniques for input features and output possibilities could be investigated to evaluate their impact on accuracy and training time.
- Feature selection: Removing feature columns with continuous data or applying advanced feature selection techniques may improve the model's efficiency, although this requires further testing and evaluation.
- Ensemble methods: Combining multiple neural network models or incorporating other machine learning algorithms (e.g., decision trees, support vector machines) in an ensemble approach could potentially improve the overall performance and robustness of the intrusion detection system.
- Transfer learning: Investigating the use of transfer learning techniques, where a pre-trained model is fine-tuned on the intrusion detection task, could potentially improve the model's performance and reduce the training time required.

- Temporal and sequential modeling: Incorporating temporal and sequential information into the neural network architecture could enhance the detection of complex intrusion patterns that evolve over time.
- Explainable AI techniques: Exploring techniques from the field of explainable artificial intelligence

6 REFERENCES

(1) KDD Cup 1999 dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
(2) KDD Cup 1999 task description: http://kdd.ics.uci.edu/databases/kddcup99/task.html
(3) YouTube video: https://www.youtube.com/watch?v=VgyKQ5MTDFc
(4) Neural network tutorial: https://sites.wustl.edu/jeffheaton/t81-558/
(5) MinMax normalization: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79
(6) Principal Component Analysis (PCA): https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
(7) Keras sequential model: https://keras.io/guides/sequential_model/
(8) Keras layer initializers: https://keras.io/api/layers/initializers/
(9) Keras activations: https://keras.io/api/layers/activations/
(10) Keras loss functions: https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function
(11) Keras optimizers: https://keras.io/api/optimizers/