



Intrusion Detection in Computer Networks Using Artificial Neural Network Approaches

Kenjiro Xanthopoulos

Department of Business Management, Kavir Polytechnic University of Technology, Iran
kenjiro.xanthopoulos@kput-ir.net

Peer Review Information

Submission: 14 Jan 2017
Revision: 01 Feb 2017
Acceptance: 17 March 2017

Keywords

Intrusion Detection System, Artificial Neural Network, Network Security, NSL-KDD, Anomaly Detection

Abstract

Intrusion detection is a critical layer of network defense because preventive mechanisms such as authentication, access control, and firewalls cannot identify every misuse or previously unseen attack. This methodology paper presents a structured artificial neural network (ANN) approach for classifying network connections as normal or intrusive. The method uses benchmark connection records, categorical encoding, numerical normalization, optional feature selection, supervised ANN training, and evaluation through accuracy, precision, recall, F1-score, detection rate, false-positive rate, and confusion-matrix analysis. The proposed design emphasizes a multilayer perceptron as a transparent baseline while also describing how recurrent or hybrid neural structures can extend the model. NSL-KDD is recommended as the primary benchmark because it reduces duplicate-record effects associated with KDD Cup 99. The method is aligned with ANN-based intrusion-detection research published through 2017. Reported benchmark evidence shows that neural models can learn nonlinear traffic patterns effectively, although class imbalance, low-frequency attacks, changing traffic distributions, and false alarms remain important limitations. The resulting workflow is suitable for a reproducible academic implementation and comparative IDS experimentation.

Introduction

Networked information systems continuously exchange large volumes of heterogeneous traffic, creating opportunities for denial-of-service, probing, unauthorized access, privilege escalation, malware delivery, and other malicious activities. An intrusion detection system (IDS) observes host or network activity and identifies events that deviate from accepted behavior or match known malicious patterns. Denning's classical intrusion-detection model established the idea that abnormal use can be recognized from monitored system behavior and statistical profiles [1]. Later benchmark evaluations showed that practical IDSs can detect many known attacks but may perform poorly on new, stealthy, or insufficiently modeled attacks

[2]. These limitations motivate adaptive methods that can learn complex boundaries from data rather than depend only on fixed signatures.

Artificial neural networks are attractive for intrusion detection because they can approximate nonlinear decision surfaces, combine many correlated traffic attributes, and generalize from labeled examples. Early work by Mukkamala, Janoski, and Sung compared neural networks with support vector machines for intrusion classification and demonstrated that data-driven classifiers can learn useful attack patterns from benchmark traffic [3]. Chen, Hsu, and Shen further compared ANN and SVM approaches using DARPA audit data, showing that performance depends not only on the

classifier but also on how security events are encoded [4]. A neural IDS therefore should not be treated as only a choice of network architecture; preprocessing, feature representation, class distribution, and evaluation protocol are equally important.

The objective of this paper is to define a concise, reproducible methodology for ANN-based network intrusion detection using publications available up to 2017. The proposed workflow focuses on supervised learning with an MLP baseline, because it is simple to implement and provides a clear reference for more advanced neural models. The methodology answers four practical questions: how network traffic should be represented, how an ANN should be trained without leakage, how performance should be measured across normal and attack classes, and how published neural IDS results should be interpreted without assuming that values obtained under different datasets or splits are directly comparable.

Literature Review

Neural-network intrusion detection evolved from basic feed-forward classifiers toward dimensionality-reduction, ensemble, online-learning, and recurrent architectures. Mulkamala et al. [3] used ANN and SVM methods to discover patterns capable of separating normal and intrusive behavior. Their work helped establish intrusion detection as a supervised classification problem. Chen et al. [4] later showed that the representation of audit information strongly influences ANN performance, reinforcing the need to design the feature pipeline and classifier together.

Dimensionality and redundancy became important issues as IDS datasets grew. Liu, Zhang, and Yang proposed a hierarchical intrusion-detection model using principal-component-analysis neural networks and reported that hierarchical processing can support both misuse and anomaly-oriented classification [5]. Tsai et al. reviewed machine-learning intrusion detection and found that a wide range of supervised and hybrid methods had been applied, but comparison was difficult because datasets, preprocessing procedures, and validation designs varied substantially [6]. This observation remains central to methodology design: an ANN result is meaningful only when the train/test split, feature encoding, attack taxonomy, and evaluation metrics are stated explicitly.

Benchmark quality is another major consideration. Tavallaee et al. analyzed KDD Cup 99 and identified duplicate-record and difficulty-distribution issues that can distort evaluation; they proposed NSL-KDD as a refined benchmark for intrusion-detection experiments [7]. This dataset has therefore been widely used to evaluate neural and non-neural classifiers. Wang et al. combined artificial neural networks with fuzzy clustering to improve stability and detection precision, especially for low-frequency attacks [8]. Govindarajan and Chandrasekaran examined multilayer perceptron and radial-basis-function networks and showed that ensemble and hybrid neural structures can improve classification compared with single base classifiers [9].

Later work emphasized efficiency and richer neural structures. Singh, Kumar, and Singla used an online sequential extreme learning machine together with traffic profiling and feature selection; on binary NSL-KDD classification they reported 98.66% accuracy and a 1.74% false-positive rate under their experimental setting [10]. Ambusaidi et al. demonstrated that removing irrelevant and redundant features can improve intrusion-detection efficiency and classification quality, particularly for high-dimensional network traffic [11]. Finally, Yin et al. introduced a recurrent neural network IDS in 2017 and reported 83.28% binary test accuracy on KDDTest+ under the stated RNN configuration, illustrating the growing interest in sequence-aware neural models [12]. Collectively, these studies support a methodology that combines careful preprocessing, explicit validation, neural classification, and class-specific performance reporting.

Methodology

1. Research Design and Dataset

The study follows a supervised experimental design. Each network connection is represented by a vector of traffic attributes and a class label. NSL-KDD is selected as the primary benchmark because it preserves the established KDD feature structure while reducing many duplicate-record problems identified in KDD Cup 99 [7]. The standard KDDTrain+ partition should be used only for model development and the standard KDDTest+ partition should be reserved for final evaluation. If cross-validation is performed, it must be restricted to the training partition so that test data never influences model selection.

Table 1. Feature groups and preprocessing used in the proposed ANN intrusion-detection methodology.

Feature group	Examples	Processing	Purpose
Basic connection	duration, protocol, service, flag, bytes	One-hot encode categorical fields; log/scale skewed numeric values	Describe each connection
Content	failed logins, root access, shell indicators	Numeric scaling; optional low-variance filtering	Capture suspicious host behavior
Traffic	count, srv_count, error rates	Standardize; preserve rate variables	Represent short-term traffic patterns
Host-based traffic	dst_host counts and rates	Standardize; feature selection if needed	Represent destination-level behavior
Label	normal or attack category	Binary or five-class mapping	Define supervised target

2. Data Preprocessing

Preprocessing is performed in the following order. First, records are checked for missing or malformed values. Second, the categorical protocol_type, service, and flag attributes are transformed using one-hot encoding; assigning arbitrary integers should be avoided because an ANN could interpret those values as ordinal. Third, continuous variables are normalized using parameters learned only from the training data. Standardization to zero mean and unit variance is suitable for most ANN inputs, while strongly skewed count variables can be log-transformed before scaling. Fourth, labels are mapped to either two classes (normal/attack) or five classes (normal, DoS, Probe, R2L, and U2R).

Feature selection is optional but should be applied when computational cost or redundant inputs are a concern. Mutual-information or correlation-based filtering may be used on the training set, following the general motivation demonstrated by Ambusaidi et al. [11]. The same selected feature indices and preprocessing parameters are then applied unchanged to validation and test data. Class imbalance should be handled through class weights or controlled resampling inside the training data rather than by altering the test distribution.

3. ANN Architecture and Training

The baseline classifier is a multilayer perceptron with an input layer matching the processed feature dimension, two hidden layers, and an output layer for binary or multiclass prediction. A practical starting configuration is 64 neurons in the first hidden layer and 32 neurons in the second hidden layer with rectified linear unit activation. Dropout or L2 regularization may be introduced when training accuracy increases while validation performance degrades. Binary classification uses a sigmoid output, whereas five-class classification uses a softmax output. Cross-entropy is minimized using a gradient-based optimizer such as Adam. The learning rate, hidden-layer width, batch size, and regularization strength are selected by validation performance rather than test performance.

Training proceeds for a fixed maximum number of epochs with early stopping. The validation loss is monitored after each epoch, and training stops when no improvement occurs for a selected patience period. Model weights from the best validation epoch are retained. To estimate stability, the complete training procedure should be repeated with at least three random seeds. This is important because neural networks can produce different local solutions even when the same dataset and architecture are used.

**Figure 1.** Proposed ANN-based network intrusion-detection architecture and processing flow.

4. Detection and Decision Logic

During detection, an incoming connection is transformed by the same encoder and scaler

used during training and then passed to the ANN. For binary classification, the output probability $p(\text{attack})$ is compared with a decision threshold.

The default threshold of 0.50 may be adjusted on validation data when the operational cost of false alarms differs from the cost of missed attacks. A higher threshold usually reduces false positives but can also reduce attack recall. The final IDS therefore records both the predicted class and the network confidence score, allowing security analysts to prioritize events without changing the underlying classifier.

5. Evaluation Measures

Performance is assessed using the confusion-matrix counts true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Accuracy is $(TP+TN)/(TP+TN+FP+FN)$; precision is $TP/(TP+FP)$; recall or detection rate is $TP/(TP+FN)$; false-positive rate is $FP/(FP+TN)$; and F1-score is the harmonic mean of precision and recall. For five-class experiments, macro-averaged precision, recall, and F1-score should be reported because overall accuracy can hide poor performance on rare R2L and U2R attacks. Training time and inference time should also be

recorded when the model is intended for real-time use.

Results

Because this paper is methodology-centered, the results are presented as reference-aligned benchmark evidence rather than as a claim that a new network deployment was executed. The literature demonstrates that neural IDS performance is strongly affected by architecture, feature preparation, dataset version, and the exact test partition. For example, Singh et al. reported 98.66% binary accuracy with a 1.74% false-positive rate for an OS-ELM-based NSL-KDD method [10], while Yin et al. reported 83.28% binary test accuracy for RNN-IDS on KDDTest+ [12]. These values should not be ranked directly because the preprocessing, training procedure, neural structure, and experimental conditions differ.

Table 2. Published neural IDS findings used to calibrate interpretation of the proposed methodology (all sources are 2017 or earlier).

Study	Neural approach	Dataset	Reported result	Methodological implication
Liu et al. [5]	PCA neural network	DARPA 1998	Satisfactory hierarchical classification (reported)	Dimensionality reduction can support online classification.
Wang et al. [8]	Fuzzy clustering + ANN	KDD Cup 99	Improved precision/stability (reported)	Cluster-aware training can help minority attack patterns.
Govindarajan & Chandrasekaran [9]	MLP/RBF ensemble	KDD benchmark data	Ensemble outperformed single classifiers (reported)	Combining neural learners can improve robustness.
Singh et al. [10]	OS-ELM	NSL-KDD	98.66% accuracy; 1.74% FPR	Efficiency and feature reduction matter for deployable IDS.
Yin et al. [12]	RNN-IDS	NSL-KDD KDDTest+	83.28% binary test accuracy	Sequential neural models are viable but benchmark difficulty remains visible.

For an implementation of the proposed MLP workflow, the primary result should be the confusion matrix on untouched KDDTest+ data, followed by attack recall, false-positive rate, macro-F1, and inference time. A credible result is not simply the highest accuracy: a model can obtain high accuracy by favoring common classes while missing rare attacks. Therefore, the methodology treats minority-class recall and false alarms as core outcomes. Repeated runs should be summarized using mean and standard deviation so that a result caused by one favorable

random initialization is not presented as stable performance.

The expected pattern from the literature is that preprocessing and feature reduction improve training efficiency, while hybrid or recurrent models can capture relationships that a shallow feed-forward ANN may miss [8]-[12]. At the same time, the difference between the strong OS-ELM result reported by Singh et al. [10] and the more modest KDDTest+ result reported by Yin et al. [12] illustrates why dataset split and model protocol must accompany every performance

number. The proposed method therefore emphasizes reproducibility over isolated accuracy claims.

Conclusion and Discussion

This paper presented a step-by-step methodology for intrusion detection in computer networks using artificial neural network approaches. The method begins with a benchmark dataset, separates model development from final testing, encodes categorical network features, normalizes continuous inputs, optionally removes redundant features, trains a regularized ANN, and evaluates the resulting classifier with class-sensitive metrics. The workflow is consistent with the historical development of data-driven intrusion detection from statistical anomaly models [1] and DARPA evaluation studies [2] to ANN/SVM classification [3], neural dimensionality reduction [5], hybrid ANN methods [8], efficient extreme-learning models [10], and recurrent neural intrusion detection [12]. The major advantage of the ANN approach is its ability to model nonlinear combinations of traffic features without manually writing attack signatures. Its main limitations are equally important: benchmark datasets may not represent current operational networks, rare attacks can be under-learned, neural outputs can be difficult to interpret, and normal traffic may change over time. Consequently, an ANN-based IDS should be retrained or adapted when traffic distributions shift, and its alarms should be integrated with logging and analyst review rather than treated as infallible decisions. Future work can extend the same methodology to recurrent networks, autoencoders, deeper architectures, online learning, or hybrid systems while preserving the same strict separation of training, validation, and testing data. Overall, the literature through 2017 indicates that ANN-based methods are a practical foundation for network intrusion detection when feature preparation and evaluation are handled carefully. The proposed methodology provides a compact framework for student-level implementation, controlled comparison of neural architectures, and reproducible reporting of detection accuracy and false-alarm behavior.

References

D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, 1987. DOI: 10.1109/TSE.1987.232894.

R. P. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-Line Intrusion

Detection Evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579-595, 2000. DOI: 10.1016/S1389-1286(00)00139-0.

S. Mukkamala, G. Janoski, and A. H. Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 1702-1707, 2002. DOI: 10.1109/IJCNN.2002.1007774.

W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of SVM and ANN for Intrusion Detection," *Computers & Operations Research*, vol. 32, no. 10, pp. 2617-2634, 2005. DOI: 10.1016/j.cor.2004.03.019.

G. Liu, Z. Yi, and S. Yang, "A Hierarchical Intrusion Detection Model Based on the PCA Neural Networks," *Neurocomputing*, vol. 70, no. 7-9, pp. 1561-1568, 2007. DOI: 10.1016/j.neucom.2006.10.146.

C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion Detection by Machine Learning: A Review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994-12000, 2009. DOI: 10.1016/j.eswa.2009.05.029.

M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1-6, 2009. DOI: 10.1109/CISDA.2009.5356528.

G. Wang, J. Hao, J. Ma, and L. Huang, "A New Approach to Intrusion Detection Using Artificial Neural Networks and Fuzzy Clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225-6232, 2010. DOI: 10.1016/j.eswa.2010.02.102.

M. Govindarajan and R. M. Chandrasekaran, "Intrusion Detection Using Neural Based Hybrid Classification Methods," *Computer Networks*, vol. 55, no. 8, pp. 1662-1671, 2011. DOI: 10.1016/j.comnet.2010.12.008.

R. Singh, H. Kumar, and R. K. Singla, "An Intrusion Detection System Using Network Traffic Profiling and Online Sequential Extreme Learning Machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609-8624, 2015. DOI: 10.1016/j.eswa.2015.07.015.

M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp.

2986-2998, 2016. DOI:
10.1109/TC.2016.2519914.
C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning
Approach for Intrusion Detection Using

Recurrent Neural Networks," IEEE Access, vol. 5,
pp. 21954-21961, 2017. DOI:
10.1109/ACCESS.2017.2762418.