# Intrusion Detection on DARPA by using Multi-layer Perceptron

Muhammed Cihat Ünal
*Computer and Artificial Intelligence Engineering*
*Hacettepe University*
*Ankara, Turkey*
*Email: muhammed.unal@hacettepe.edu.tr*

***Abstract*—Day by day, computer networks are getting wider thanks to new technologies, social media applications, computer games and so on. This leads huge data interaction among the systems. Enormous data are collected, analyzed and stored in even a single day by many companies and systems. Consequently, security has become one of the biggest vital issue for computer systems. This paper presents neural network based approach for intrusion detection while classifying the attacks. One of the most important reasons why this study is valuable that the model not only detects whether the attack exist, but also it detects the type of the attacks. In other words, the presented approach aims to solve multi-class problem by using Multi Layer Perceptron (MLP). Various neural architectures have been tried to find optimal model. The results show that even simpler neural networks are capable of detecting the attacks with high accuracy.**

***Index Terms*—Artificial Neural Networks, Multi Layer Perceptron, Intrusion Detection, Multi-Class Classification**

## 1. Introduction

Computing worlds are changing rapidly thanks to expansion of World Wide Web and local network systems. As the network spreads among the systems all around the world, it leads both beneficial and hazardous consequences. The developments in social networks, artificial intelligence, automotive industry, etc. can be defined as beneficial developments. However, even in these areas, new and equipped intruders and hackers emerge alongside with their calamitous purposes. The costs of temporary or permanent damages caused by unauthorized access of the intruders to computer systems have urged different organizations to increasingly implement various systems to monitor data flow in their networks [1]. These systems are called Intrusion Detection Systems (IDSs). During the past few years, many organizations have exposed sophisticated cyber-attacks, and thereby needs for robust and innovative IDSs elevated. The development of IDSs concerns both the academic and the industrial community worldwide due to the impact that each cyber attack has, as economic cost, reputational damage, and legal sequences [2]. Therefore, securing the networks to prevent unauthorized access and violation of the private data has great importance.

There have been approaches to mitigate this problem, but one of the common approach is Denning's profile model[3]. Rule-based analysis relies on sets of predefined rules, nonetheless it is impossible to define every rule, since attack types are vary, and too hard to do frequent update to keep across new attacks. These detection designs suffer from inflexibility because if the sequence of events is even slightly different from the predefined rules.

## 2. Literature Review

As the importance of computer security grows, more approaches are proposed to overcome this problem. Since DARPA dataset includes many labelled attack scenarios, there are many trials have been performed on this dataset by using different neural architectures and solutions.

In [4], the authors performed experiments on DARPA dataset multi-layer perceptron (MLP). In their experiment, they've tried different configurations, varying the number of hidden layers and the number of training epochs to obtain optimal model for intrusion detection. They use a widely-used open-source platform KNIME for data analysis, feature extraction and training. Unlike our experiments, they used all the features in the training. At the end, they argue that MLPs are efficient models for classification tasks since they've achieved high classification rate with low error. Nevertheless, there are couple deficiencies that make this experiment unreliable as they conclude all experiments on KNIME and didn't share much information about train, test and validation data, success rate of the model, training time and optimal parameters.

The study in [5] proposes a new intrusion detection technique by combining K-Means and MLP. Specifically, samples are divided into sub-samples by using K-Means, and then each sub-sample is fed into Neural Network. Thus, they aim to avoid from slow convergence and computational burden of NNs. They focus on sampling and changes in neural network rather than preprocessing along with the experiments. The experiments were conducted using 1998 DARPA BSM audit data. They employ three hidden layers in their network with varying hidden nodes and one neuron in the output layer, as a binary classification task.

The paper [6] introduces an anomaly intrusion detection approach based on a Hybrid Multi-Layer Perceptron (MLP) and Chaotic Neural Network (CNN) to address the limitations of existing MLP-based intrusion detection systems. Specifically, the focus is on enhancing the detection of time-delayed attacks, a type of attack that conventional neural network Intrusion Detection Systems struggle to identify efficiently. Tests are conducted using the DARPA 1998 dataset, and the experimental results are presented and compared using ROC curves.

## 3. Methodology

In this section, information will be provided about the evaluation dataset, preprocessing steps and neural network architectures, which were used in this experiment, will be discussed.

### 3.1. Dataset

The experiments in this study have been concluded on 1999 Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation dataset. The raw version of that contains more than 450,000 connection records, but 20,055 records are included in [7]. The authors stated that they created this subset manually by choosing a reasonable number of normal events and attack types arbitrarily. We've followed the same strategy in this study, but the authors didn't give the details about how they chose the data. Therefore, we've taken the 20,055 records randomly from the raw dataset.

There are at least four different known categories of computer attacks including denial of service attacks, user to root attacks, remote to user attacks and probing attacks [8][7]. Sixty different attack types exist in 1999 DARPA dataset, however only two of them, which belong to different attack categories, used in this experiment: SYN Flood (Neptune) and Satan. These attack types are under Denial Of Service and Probing attack categories respectively. The authors preferred to use these attack types due to the availability of enough data and the possibility to compare with previous works that use the same types. Table 1 shows detailed information about the amount of attack types in train, validation and test datasets. In the following subsections, a description of the attack types is provided.

**3.1.1. SYN Flood.** A SYN flood is a type of Distributed Denial of Service (DDoS) attack in which initial connection request (SYN) packets are sent to overwhelm all available ports on a target (victim) so that it is unavailable to legitimate traffic.[9]

**3.1.2. Satan.** Satan is a probing intrusion which automatically scans a network of computers to gather information or find known vulnerabilities. The network probes are quite useful for attackers planning a future attack.[8]

TABLE 1. DATA DISTRIBUTIONS OF ATTACK TYPES IN TRAIN, VALIDATION AND TEST SETS.

| Record Types | Training Set | Validation Set | Test Set |
|---|---|---|---|
| Normal | 5,922 | 300 | 3,608 |
| SYN Flood | 4,430 | 300 | 2,321 |
| Satan | 1,807 | 300 | 1,067 |

### 3.2. Feature Selection, Numerical Representation, and Normalization

There are 41 features in the DARPA dataset. 22 of these features describe the connection itself and 19 of them describe the properties of connections to the same host in the last two seconds [7]. In various attack scenarios, the signature of the attack record is determined by scrutinizing specific attributes within a sequence of records. Consequently, the Intrusion Detection System (IDS) must assess the service types employed by a user in prior connections. To achieve this objective, a feature vector encompassing 19 features that characterize past events in the computer network has been incorporated. All the the features in the dataset were described in [7] by grouping them into four titles.

In [7], it was stated that values of 6 columns (land, urgent, num_failed_logins, num_shells, is_host_login, num_outbound_cmds) are constantly zero. These columns (features) have no effect on the prediction of the model since they don't provide any discriminative knowledge. They are time consuming, and they only make things complicated. Therefore, these features have been subtracted from the dataset. Three features (protocol_type, service, flag) were non-numerical. This is undesirable since the feature vector fed to the input of the neural network has to be numerical. We've followed the same mapping strategy to make protocol types numerical as described in [7]: tcp=0, udp=1, icmp=2. Nevertheless, similar instructions were not given for other 2 columns, so we converted them arbitrarily.

### 3.3. Neural Network Architectures

The features reside in disparate ranges, making them non-comparable. While certain features adopted binary values, others spanned a continuous numerical range. Consequently, to address this discrepancy, normalization was employed. This involved mapping all distinct values for each feature to a standardized range [0, 1].

The current study focuses on multi-class classification since we have two different attack types and one normal event. Thus, we need to have 3 neurons at the output layer of our Multi Layer Perceptron. The authors don't mention about activation function they used in the output layer, but as far as we understand, the sigmoid function has been used instead of softmax activation because they aim to get [1 0 0] for normal conditions, [0 1 0] for Neptune attack and [0 0 1] for the Satan attack. However, there is a possibility that the output of a neural network can provide results like [1 1 1], hence these are considered irrelevant cases.

To imitate the experiment in [7], two different MLP networks are presented, and three experiments have been concluded. First neural architecture has one input layer, two hidden layers and an output layer at the end. This architecture has been tested using 35 dimensions (same with the dimension of the feature vector) in both input and hidden layers. So, the overview of the neural architecture in terms of neurons is {35 35 35 3}. The second neural architecture has two differences from the first architecture: 1 hidden layer and 45 neurons in the hidden layer. Two different experiments occurred with the second architecture. In the first experiment, the hidden layer was reduced to one, but neurons remained the same {35 35 3}. In the second experiment, the neuron size was increased to 45 {35 45 3}.

## 4. Experiments And Results

From now on, the experiments in [7] will be entitled as Original Experiment.

The authors use three different methods for doing original experiments. Firstly, they train a neural network, which is addressed in 3.3, with 200 epochs, and then they assess the mean squared error during the training, train-test accuracy score and training time. Secondly, they repeat the same experiment by adding an early-stopping method, and then compare the evaluation metrics, that are used in the first step of the training. Lastly, they repeat the same methods in the second experiment by reducing the hidden-layer from two to one.

As can be seen in Table 2, the model in the original experiment was overfitted, as pointed out in [7], however, this was not the case in our experiment. Although the error graph 1 is similar with the original experiment, our model's generalization is better and not overfitted according to our observations. Execution times of training between our training and original training differ tremendously. We've done the experiments on CPU, but hardware resources weren't given for original experiments in [7].

TABLE 2. COMPARISON OF FIRST EXPERIMENT RESULTS

| Experiment | Train Acc. | Test Acc. | Execution Time |
|---|---|---|---|
| Original Exp. | 1̃00% | 8̃0% | +25 hrs. |
| Imitation Exp. | 95.9% | 91.7% | 12.9 sec. |

For the second experiment, early-stopping method has been added to model training with patience as 10. In that implementation, training will stop if the error increases ten times consecutively. The authors benefit from this implementation; training has stopped in 45th epoch, thus execution time shortened (from 25 hrs. to 5 hrs.), training accuracy dropped slightly (from 100% to 98%) while test accuracy increased to 90%. In our experiment, training was never restricted by early stopping since the loss never increased consecutively even though we used 3 as patience. Therefore,

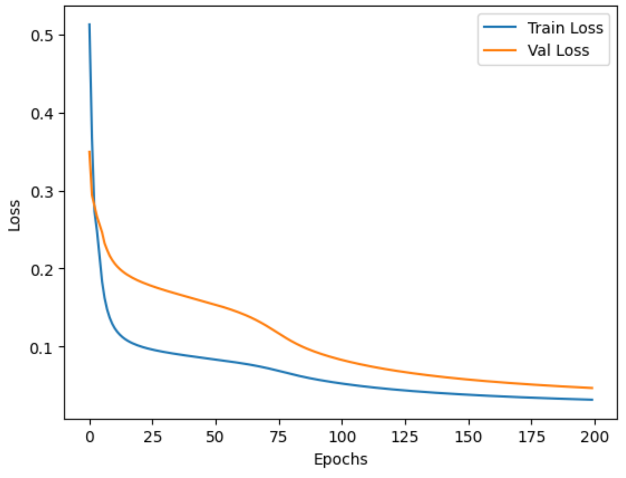

Figure 1. Means Squared Error at each epoch for first experiment.

we cannot make a comparison for this part of the experiments.

Lastly, we've imitated the third experiment. In Original Experiment, training has stopped in 48th epoch. The correct classification result was 93.1% on training and 87% on unseen data(test set). In our case, the training was completed in 200 epochs since loss never increases consecutively, the correct classification result was 95.8% on training and 87.6% on test set, and the execution time 7.95 seconds. Our model's success rate remained the same for training, but it diminished by 4% for unseen data. Execution time also decreased in our experiment. Table 3 provides a general overview for comparison.

TABLE 3. COMPARISON OF THIRD EXPERIMENT RESULTS

| Experiment | Train Acc. | Test Acc. | Execution Time |
|---|---|---|---|
| Original Exp. | 93.1% | 87% | Not clarified |
| Imitation Exp. | 95.8% | 87.6% | 7.95 sec. |

## 5. Conclusion and Discussion

As aimed, we could imitate some of the results obtained in the original experiment, but not all of them. In the first step of the experiment, there was a huge difference between experiments even though we used the exact parameters used in the original experiment. This is reasonable due to several reasons. One reason is random initialization in weights. Since we have no seed value to obtain similar weight initialization, train-test results differ. Another reason is the lack of hyperparameter knowledge in the original experiment. Such as the activation function between hidden layers and the learning rate are not given. Last, even though train-validation-test sample sizes are aligned with the original experiment, data are not the same because they choose their data manually and don't explain how to do this. Although

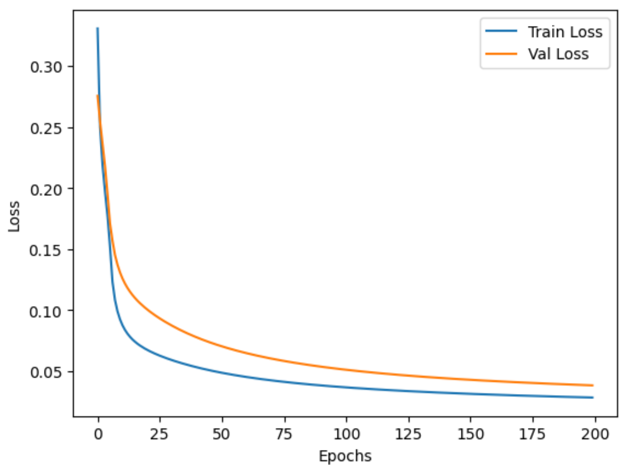

Figure 2. Means Squared Error at each epoch for third experiment.

this is the case and the shape of the loss functions are similar, the generalization was better in our experiment.

They added early stopping as a second step for improvement, but it did not affect our experiments. Again, it can be due to random weight initialization and different data distribution. In fact, I tried the training with several seed values to make results alike with original experiments, which I couldn't achieve.

In the third step of the experiment, the hidden-layer size was reduced to one and neuron size was increased to 45. We obtained nearly the same train-test accuracy. Besides, execution time decreased and no severe generalization loss occurred as intended in the original experiment.

Lastly, there is a huge difference between the execution times of experiments. This most probably occurs due to improvements in the computational powers of computers during the past years. We've concluded the experiment after 20 years from the original experiment. Hence, excessive difference in execution time is expected and comprehensible.

## References

[1] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, no. 4, pp. supl27–supl30, 2002.

[2] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1253–1258.

[3] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[4] F. Amato, N. Mazzocca, F. Moscato, and E. Vivenzio, "Multilayer perceptron: an intelligent model for classification and intrusion detection," in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2017, pp. 686–691.

[5] H. Zheng, L. Ni, and D. Xiao, "Intrusion detection based on mlp neural networks and k-means algorithm," in *Advances in Neural Networks–ISNN 2005: Second International Symposium on Neural Networks, Chongqing, China, May 30-June 1, 2005, Proceedings, Part III 2*. Springer, 2005, pp. 434–438.

[6] Y. Yao, Y. Wei, F.-x. Gao, and Y. Ge, "Anomaly intrusion detection approach using hybrid mlp/cnn neural network," in *Sixth international conference on intelligent systems design and applications*, vol. 2. IEEE, 2006, pp. 1095–1102.

[7] M. Moradi and M. Zulkernine, "A neural network based system for intrusion detection and classification of attacks," in *Proceedings of the IEEE international conference on advances in intelligent systems-theory and applications*. IEEE Lux-embourg-Kirchberg, Luxembourg, 2004, pp. 15–18.

[8] K. K. R. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[9] J. Aiken and S. Scott-Hayward, "Investigating adversarial attacks against network intrusion detection systems in sdns," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2019, pp. 1–7.