CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
https://cajmtcs.casjournal.org/index.php/CAJMTCS
Volume: 07 Issue: 02 | April 2026 ISSN: 2660-5309

Article

Optimized Deep Neural Network to Optimally Classify Attacks as an Intrusion Detection System

Ahmed Mahmood Khudhur [1]

1. Information Technology Department, College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, Iraq

* Correspondence: Dr.ahmedm@uokirkuk.edu.iq

Abstract: The expansion of internet usage has made networks increasingly vulnerable to continuous and escalating breaches. This is due to the critical nature of the information exchanged across these networks. Given the importance of this information, a method for protecting it from breaches is essential. Several such methods exist, including those within machine learning (ML) and deep learning (DL). In this research, we will utilize the Genetic Algorithm (GA) to determine the optimal hyperparameter values for the deep learning model, which will then be used to classify attacks. Results show that the accuracy was comparable to other works, 98.54%, with a precision of 98.55%. The proposed deep neural network is based on convolutional neural network (CNN). On the other hand, the suggested model will need, of course, to a training/testing dataset, thus, the NSL-Kdd dataset was affordable, where it was cleaned and prepared for the training and testing purposes. Last but not least, it is recommended to make use of the suggested approach to be generalized against attacks in real-world systems.

Keywords: convolutional neural network; genetic algorithm; machine learning; intrusion detection systems, NSL-KDD dataset; optimized deep neural network

Citation: Khudhur A. M. Optimized Deep Neural Network to Optimally Classify Attacks as an Intrusion Detection System. Central Asian Journal of Mathematical Theory and Computer Sciences 2026, 7(2), 113-123.

Received: 10th Des 2025
Revised: 11th Jan 2026
Accepted: 15th Feb 2026
Published: 02th Mar 2026

1. Introduction

The development of science and technology has laid the foundations for new applications and fields of study, such as wireless sensor networks, including those used underwater (acoustic). Furthermore, machines now communicate with each other through peer-to-peer communication. There are also communication technologies known as thing-to-thing networks. All of this has led to the emergence of the Internet of Things (IoT). The IoT itself has then established the foundations for new systems-level applications. From the above, one can imagine the volume of information circulating on the Internet, and one can understand the seriousness and importance of this information if we know that important security organizations, hospitals, factories and other organizations also use the Internet to transmit their data. Therefore, internet networks (in general) have become vulnerable to hacking. These networks must be protected with up-to-date methods because hackers are constantly evolving and updating their techniques.

There are many optimization methods for solving the problem described above. For example, some methods are inspired by natural phenomena, while others rely on pure mathematical theories. Still others are based on human neurons, which are generally called neural networks. The most effective approaches in the field of Intrusion Detection Systems

(IDSs) are those based on Deep Neural Networks (DNNs). For instance, [1] employs the self-taught learning method to construct an IDS. That is, authors in [1] have reduced the dimensionality and taught the suggested model, which is consisting of DNN (based on autoencoder-mechanism), and Support Vector Machine (SVM) for the classification purposes. However, the authors also compared their work to other approaches that based on SVMs. Their results show that the autoencoder has accelerated the SVM performance. Safaldin et al in [2] employed the binary grey wolf optimizer [3] (BGWO), a modified version, in order to improve the IDS functioning. The authors suggested to integrate BGWO with SVM. The first part, BGWO, is to reduce the dimensionality (or feature reduction), while the second part is to perform the classification job. Furthermore, the utilized dataset was NSL-KDD [4]. Evaluation metrics were, at most, accuracy, recall, and f1-score. That is, authors in [2] did not make use of the DNN algorithms.

To reduce the dependence of utilizing real datasets to train a DNN model, i.e., fake data generation to reduce the deployment of physical sensors, authors in [5] have proposed to utilize the Conditional Generating Adversarial Network (CGAN). The utilization of CGAN was enabled the authors to perform unsupervised learning. The classifier was eXtreme gradient boosting (XGBoost) [6]. NSL-KDD dataset was employed to train and test the suggested model. Evaluation parameters were just like those used in [1, 2], accuracy, recall, f1-score, as well as the precision. Since, NSL-KDD dataset is a comprehensive one and it contains 41-features/attributes, it is essential to reduce this number of features. Thus, in [7], the differential evolution methodology was utilized to achieve this target. Then different classifiers were conducted to classify the attacks, such as SVMs, Decision Tree (DT)[8], Ensembles Tree (ET), and Naïve Bayes (NB) [9]. Ensemble method was other approach to reduce the number of features, as was conducted in [10], were NSL-KDD dataset was also used. Moreover, Chameleon Swarm Algorithm [11] also employed to reduce the number of features. One can refer to [12] for further information of the machine learning algorithms that were utilized in IDSs.

Generative Adversarial Network, which was employed in [5], was made convolutional in [13], thus, providing Deep Convolutional GAN based IDS. The proposed system, although it was more complex, but it gives a comparable result in terms of accuracy, recall, f1-score, and precision. On the other hand, another dataset was employed other than the NSL-KDD. Furthermore, spherical graph Convolutional Neural Network (CNN) based DNN was integrated with planet optimization method to classify attacks in NSL-KDD and UNSB-NB15 datasets in [14]. The authors utilized clusters to reduce the noise and synthesis minority over samples in order to achieve balancing among the classes. The Improved Zeiler and Fergus Network with Dilated Residual Network was involved to extract the features, other than feature selection approaches used which is achieved using Hybrid Parrot with Addax Optimization methodology. In other words, the work included feature extraction and feature selection/reduction, therefore, leading to more complication to the suggested methodology in [14]. Phalaagae et al integrated the CNN with Long-Short Term Memory (LSTM), in order to improve feature extraction both in pattern forms and temporal format, then an improved (PSO) [15] was put to reduce the number of features. In other words, the dimensionality reduction was employed as well in the work using the PSO. Thus, a hybrid structure based on CNN and LSTM is employed, which leads to more complication of the suggested model. NSL-KDD dataset was also employed to train the model.

Authors in [16] utilized a combined grasshopper optimization algorithm and sine–cosine algorithm to conduct feature selection, i.e., dimensionality reduction, then attack detection was performed by a random Neural Network (NN) alongside grasshopper algorithm, which is employed in this stage to adjusts the structure and parameters of the random NN. Furthermore, authors of [17] employed another type of GAN which is progressive GAN with self-attention. The utilized dataset was WSN-DS [18]. In order to reduce the number of features, war strategy algorithm [19] was employed. Optimizing the

network hyperparameters was conducted through Namib Beetle Algorithm [20]. Last but not least, authors in [21] proposes to employ Kolmogorov Arnold Network (KAN) based DNN to construct an improved IDS. In fact, the Arnold network was the replacement of the fully connected/dense-layer of the DNN model. Principal Component Analysis (PCA) [22] was used to extract the features that fed to the graph CNN layers. More in depth survey about IDSs can be further read by readers to gain more knowledge about ML algorithms and DNN models that employed to build IDSs in [23].

Motivated by [16] and [17], the suggested methodology of this work is based on genetic algorithm for hyperparameters optimization and based on the results the DNN, that is based on CNN, will be structured. The NSL-KDD dataset will be used to train the suggested system. The rest of this paper is organized as follows: the dataset will be presented in section II alongside the preprocessing such as cleaning and data exploratory analysis, section III will show the suggested methodology that is based on CNN, which will be optimized using genetic algorithm, section IV will discuss the simulation and results, then the conclusion will be drawn in section V.

2. Dataset And Preprocessing

Before the production of NSL-KDD dataset, a previous version was called KDD Cup99 dataset, which is produced for the first time by researchers at the University of New Brunswick. The redundant records in KDD Cup99 and duplications alongside other issues in this version, other researchers at the University of New Brunswick have improved it to produce the NSL-KDD dataset, in order to evaluate the networks in terms of attacks. Thus, the improved version, NSL-KDD, has no redundant records, it was balanced in terms of difficulty, and acceptable/reasonable size, in both training and testing files. That is, each record has 41-attributes/features besides the label column, then, there are 42-attributes including the label that shows the connection type, such as normal connection or an attack. Various attacks are labeled in the label-attribute, i.e., different attack categories are recognized. These attacks are Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). However, the features maybe filled in numeric values or binary values or nominal values. The combination of these values leads to classify the record to either one type of the above attacks or normal communication. Because of there is no duplication in the training-set of the NSL-KDD dataset, the utilized classifier will not tend to the higher frequency records, i.e., will not be biased as in KDD Cup99.

As the percentage of records in the original KDD dataset goes increased, the number of records chosen from each difficulty level group also goes up. Because the classification rates of different ML-algorithms may be more different, it may be easier to compare different learning strategies. That is, Table 1 lists the 41-features showing their types and values, such as minimum and maximum values in the entire records of the NSL-KDD dataset. For instance, in Table 1, there are three features that are categorized as nominal: Protocol_type, Service, and Flag. Other sorts are all of type double. A feature with maximum value, 1.38×109 and zero value as minimum called src_bytes is the attribute that shows higher maximum value. There are also two further attributes in the NSL-KDD dataset: "Label" and "Level." The Label element will inform you if a record is being attacked, and the Level attribute will tell you how bad the attack is (i.e., the severity-level).

Tabel 1. NSL-KDD dataset features description

No.	Feature	Type	Min	Max
1	Duration	double	0	42908
2	Protocol_type, Service, and Flag		Nominal	
3	src_bytes	double	0	1.38×10^{9}
4	dst_bytes	double	0	1.309×10^{9}
5	dst_host_srv_serror_rate,Land, logged_in, root_shell,is_host_login,is_guest_login, serror_rate, srv_serror_rate, rerror_rate, dst_host_rerror_rate,srv_rerror_rate,dst_host_same_srv_rate, dst_host_srv_rerror_rate,srv_diff_host_rate, dst_host_same_src_port_rate, diff_srv_rate, dst_host_serror_rate,dst_host_srv_diff_host_rate, same_srv_rate, and dst_host_diff_srv_rate	double	0	1
6	wrong_fragment and urgent	double	0	3
7	Hot	double	0	77
8	num_failed_logins	double	0	5
9	num_compromised	double	0	7479
10	su_attempted and num_shells	double	0	2
11	num_root	double	0	7468
12	num_file_creations	double	0	43
13	num_access_files	double	0	9
14	num_outbound_cmds	double	0	0
15	Count	double	0	514
16	srv_count	double	0	511
17	dst_host_count and dst_host_srv_count	double	0	255

As mentioned above, there are numerous attack sorts, but in this paper, binary classification will be conducted. In other words, DoS, R2L, U2R, and probe attack types will be all re-labeled under one-category called attack. The dataset, i.e., NSL-KDD, will be first checked-out in terms of missing values, redundant, and not logically filled values. Moreover, constants all over the recodes that are constant with identical values will be removed-out. On the other hand, nominal features may include: icmp, tcp, and udp for the Protocol_type attribute, http, private, domain_u, others are all sorted under the service-attribute. Moreover, S0, S1, S2, S3, SF, SH, OTH, REJ, and RSTO are under Flag-attribute. All of these attributes will be encoded to numerical values in order to enable the optimization algorithm to be trained then tested

3. Suggested Methodology

The suggested model is consisting of 9-layers including the input and output layers. The output layer is simply a dense-layer with one node, since the required output is either normal flow or an attack. The input layer will be optimized with the other layers using the genetic algorithm, as will be described consequently. However, the general structure of the proposed DNN based on CNN can be seen in Figure 1.

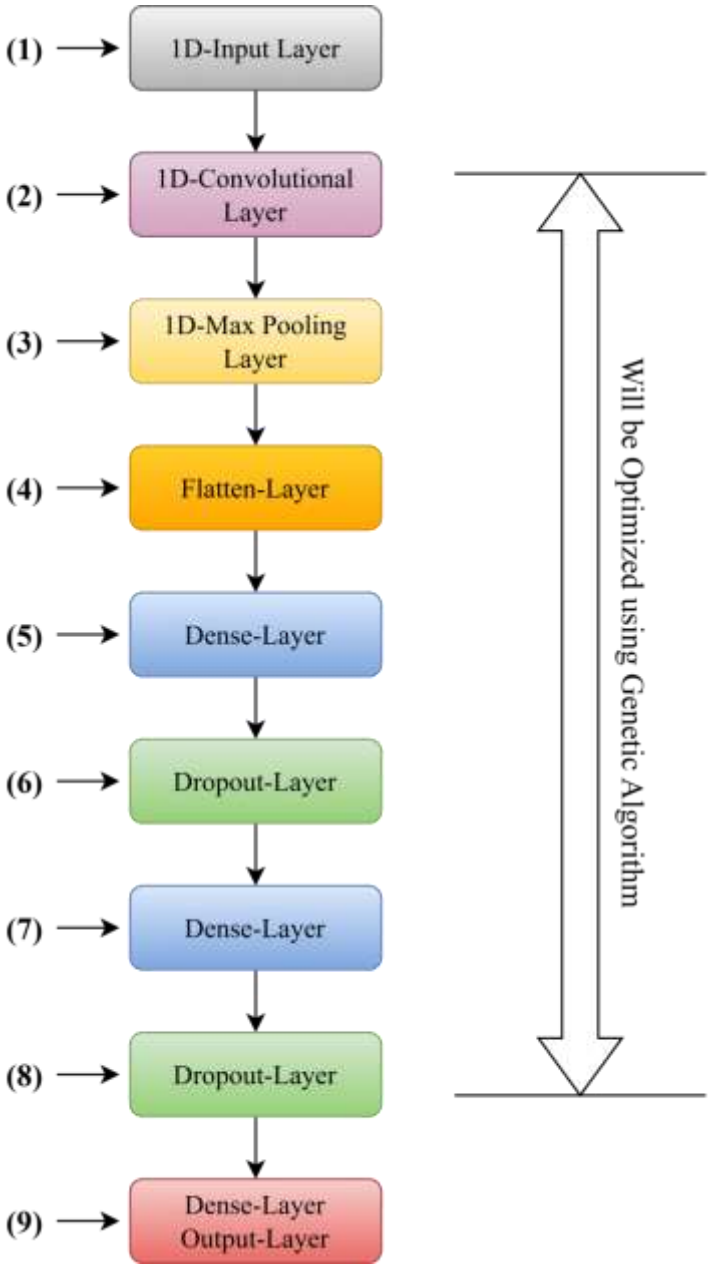

Figure 1. General architecture of the suggested model

The general structure of Figure 1 should be optimized, in terms of hyperparameters. The genetic algorithm will run for 5-cycles or it can be said there are 5-generations. The population size will be 10, i.e., the number of hyperparameter sets per generation will be 10. Thus, each generation, the algorithm tests 10-different hyperparameter combinations. Though, there will be 50-different models that the model trains before deciding the best model hyperparameters. At the end, genetic algorithm will produce the best combination of hyperparameters: first 1D convolutional layer-filters, first 1D convolutional layer-kernel size, dense layers units (two layers), and the dropout rates. No more discussion can be given here, since the results of the genetic algorithm are considered in the next section. That is, the next section will show the results of the genetic algorithm generations, the final structure of the proposed model in this section, see Figure 1, training, testing, and analysis of the model.

4. Results And Discussion

Based on the suggested methodology in the previous section, the DNN based CNN model can be imagined as listed in Table 2. Hence, there will be 82,241 trainable parameters. Thus, there are 12-trainable parameters in the first convolutional layer, 77952-trainable parameters in the first dense layer, 4128-trainable parameters in the second dense layer, and only 33-trainable parameters in the last (output) dense layer. There are zero-trainable parameters in all of the other layers. In other words, the max-pooling layer, flatten layer, and dropout layers are all have zero-trainable parameters.

Tabel 1. Optimized model result

No.	Layer	Optimization Results
1	1D Convolutional Layer	(32, 3) (No. Filters, Filter size)
2	1D Max Pooling layer	2 (not optimized)
3	Flatten Layer	608 (not optimized)
4	Dense Layer	128
5	Dropout Layer	0.08
6	Dense Layer	32
7	Dropout Layer	0.089
8	Dense Layer (output)	1 (not optimized)

However, the number of filters, 32, can be considered as a good starting point where the kernel size, 3, small enough to be able to capture local patterns. Noting the first dense layer with 128-units is fairly large but the second dense layer has only 32-units (making something like the bottleneck layer). Hence, this behavior is like a funnel: 128 → 32, that will give good capacity to the model to learn hierarchical characteristics. On the other hand, the dropout layer is dropping relatively small amounts, 8.9%, this suggests that the model do not requires heavy regularizations and the data is not prone to severe over-fitting.

Consequently, the optimization will start with the fixed learning rate of magnitude 0.001, while the optimizer itself will be "Adam". The batch size is also fixed to 32 and 100-epochs. Dataset was split into 70% for training and 30% for the testing phase. The simulation was conducted using Kaggle.com framework, since it provides free frameworks to users across the world. According to the settings above, simulation performed and Figure 2 shows the training and testing phases.

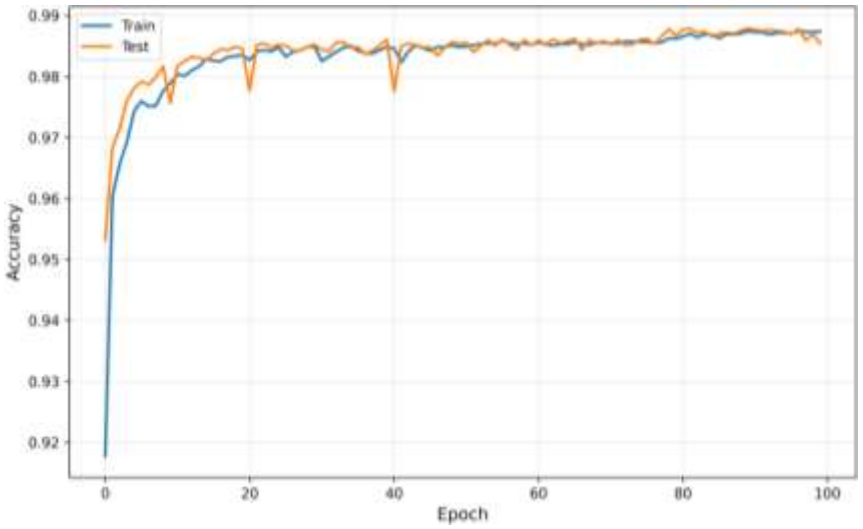

Figure 2. Training and testing accuracy phases of the proposed model

It is observed that the two curves, of the training and testing, are almost comparable, or it can be said they are matching to each other, this refers to that the dataset do not makes

the model goes to over/under-fitting problems. This is a confirmation to the genetic algorithm when it found that the dropout rate to be 8.9% only, while the common rates are between 20%-to-50%. Accordingly, the training accuracy was 98.66%, while the testing accuracy is 98.54%. Thus, the gap between them is 0.12%, i.e., an ignorable gap between the training and testing phases, this discussion suggests low-level of overfitting and the model could be generalized. During simulation, the author has put a programmed gauge that shows the overfitting issue, as indicated in Figure 3, where it can be noticed that the gauge indicated low-overfitting level.

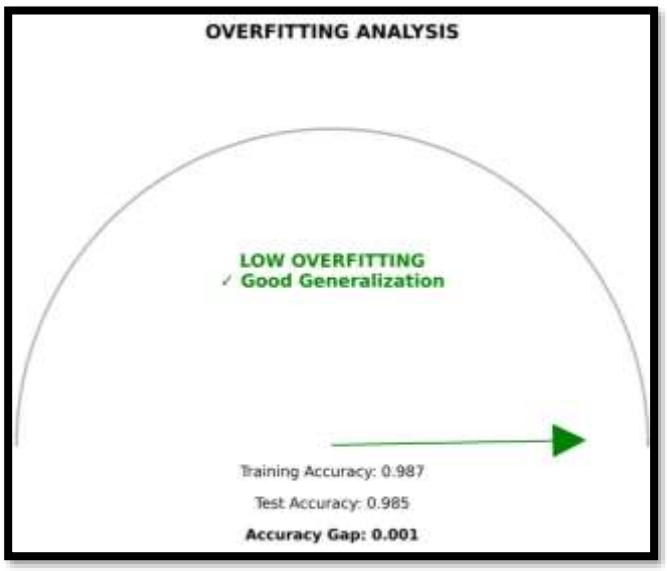

Figure 3. Over/Under-fitting gauge indicator showing a very low accuracy gap

Nevertheless, Figure 4 shows the losses of the training and testing phases. The same attitude of the performance in Figure 2 was captured in Figure 4, where almost the two phases, train and test, are matching, indicating a second confirmation to the results of the optimized parameters using genetic algorithm. By comparing the two performances in Figures 2 and 4, it can be observed that the suggested model is performing well without suffering from over/under-fitting issues.

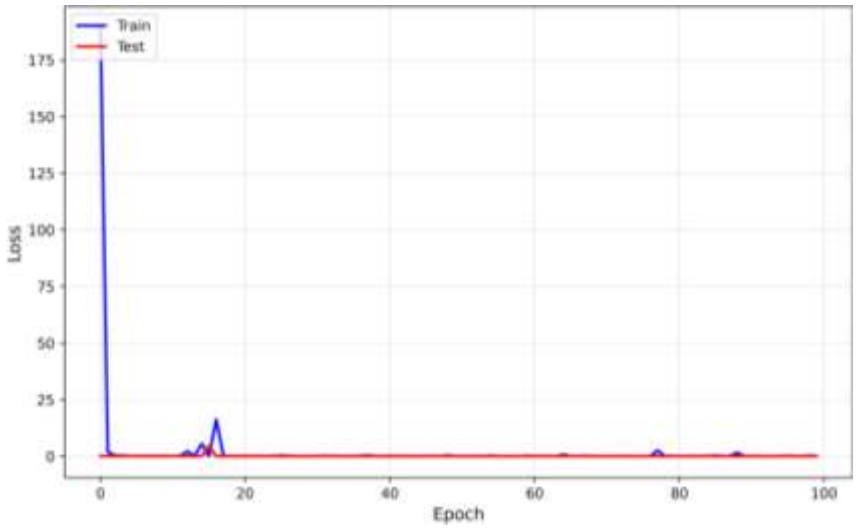

Figure 4. Training and testing losses phases of the proposed model

The confusion matrix in Figure 5 could show various metrics that can be used to evaluate the model. For instance, the accuracy, recall, precision, and F1-score. It is essential

to note that Figure 5 was drawn for the training data, this stands for that the model is already saw this data before, that is why the percentage values in the figure are excellent.

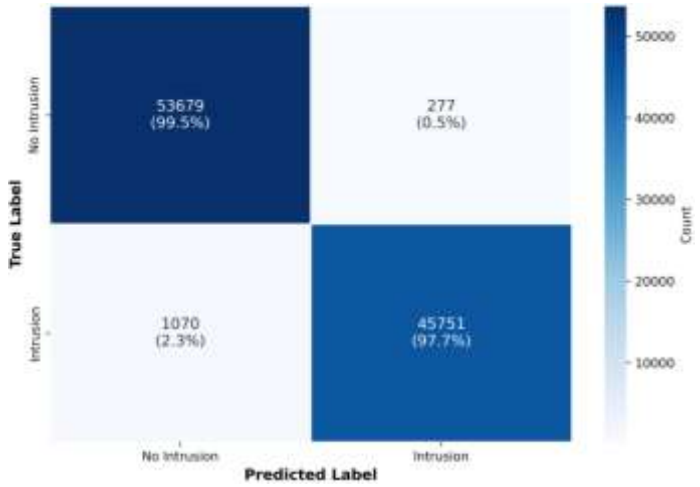

Figure 5. Training phase confusion matrix of the proposed model

The testing phase confusion matrix is shown in Figure 6. The numbers are also interesting. Note that the differences between the values between Figure 5 and Figure 6 are too small (see Figure 7). This confirms that the system did not overfit during training. Indicating the optimization process that was achieved using genetic algorithm was good.

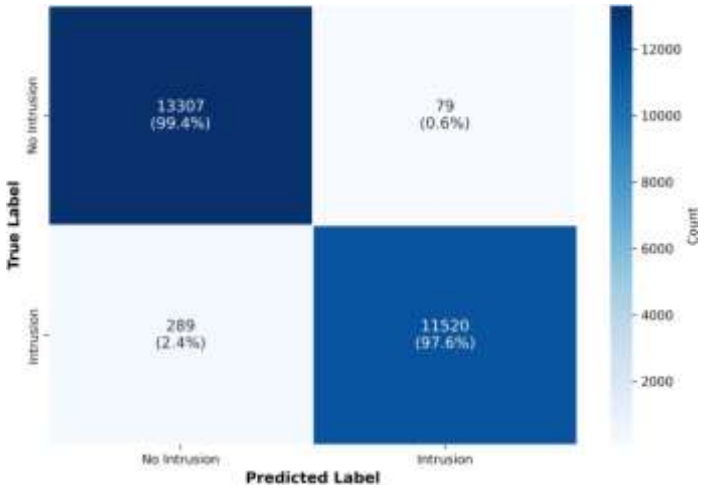

Figure 6. Testing phase confusion matrix of the proposed model

Consequently, the best accuracy achieved by the trained model was 98.66% for the training-phase and 98.54% for the testing-phase, i.e., there is a difference as a little as 0.12%. the precisions are 98.67% and 98.55% for the training and testing phases, respectively, with a gap of 0.12% between them. The recall performance metric shows 98.66% in the training phase and 98.54% in the testing phase with 0.12% of gap between the training and testing phases. Last but not least, F1-score results were captured as 98.66% in the training phase and 98.54% in the testing phase, as shown in Figure 8.

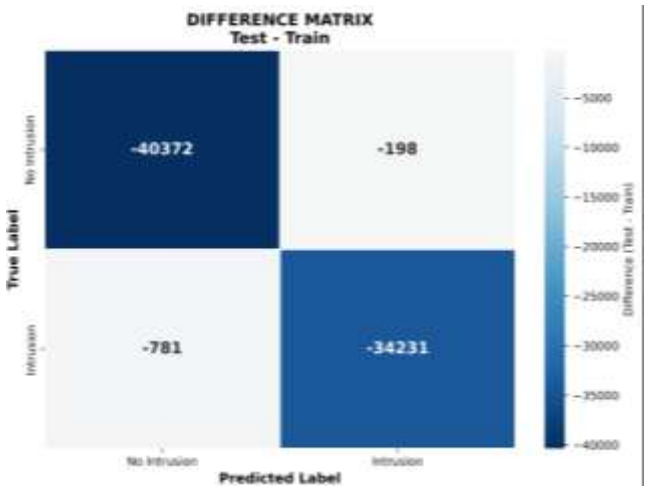

Figure 7. Training-Testing difference confusion matrix

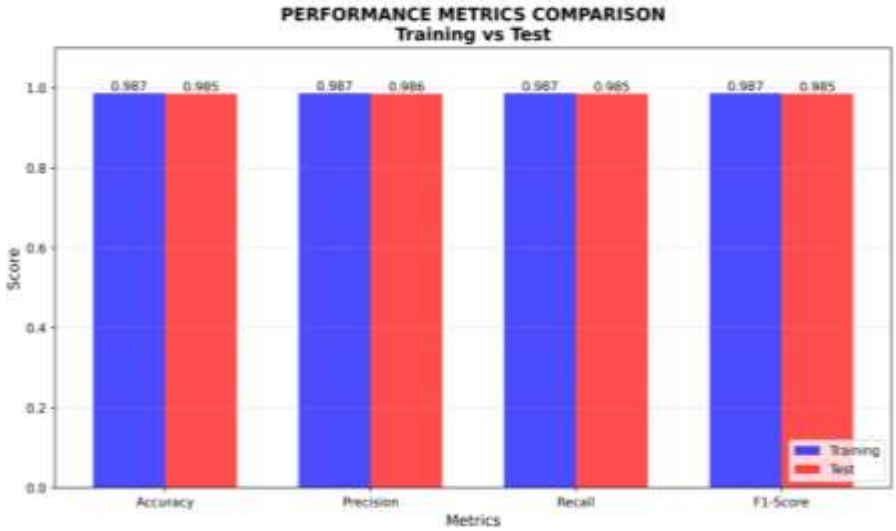

Figure 8. Training-Testing performance metrics drawn from the train/test-confusion matrices

Consequently, as per class specific metrics (in the testing-phase), the non-intrusion precision, recall, F1-score, and specificity are 97.87%, 99.41%, 98.64%, and 99.41%, respectively

5. Conclusion

The research works to employ the genetic algorithm aiming to do hyperparameters optimization of the deep neural network that is based on the convolutional neural network (one-dimensional, 1D). Thus, automatically, the genetic algorithm will determine the optimal values for the number of filters/kernels and its size. Moreover, the total number of nodes in the fully connected layers (dense-layers). Furthermore, the dropout rate for the two layers in the suggested model. In other words, the best-performing configuration was then evaluated on unseen test data to assess its generalization capability. The obtained hyperparameters indicate a balanced model structure with the aim of maintaining the less model complexity. Consequently, the optimized structure of the suggested model has shown a comparable performance in both subsets, training and testing. That is, the small gap between the accuracy, recall, F1-score, and precision evaluation metrics indicates that the optimized model can be generalized with ignorable or even neglected overfitting or underfitting issues.

REFERENCES

[1] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection," IEEE Access, vol. 6, pp. 52843-52856, 2018, doi: 10.1109/ACCESS.2018.2869577.

[2] M. Safaldin, M. Otair, and L. Abualigah, "Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 2, pp. 1559-1576, 2021/02/01 2021, doi: 10.1007/s12652-020-02228-z.

[3] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-Verse Optimizer: a nature-inspired algorithm for global optimization," Neural Computing and Applications, vol. 27, no. 2, pp. 495-513, 2016/02/01 2016, doi: 10.1007/s00521-015-1870-7.

[4] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 Military Communications and Information Systems Conference (MilCIS), 10-12 Nov. 2015 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[5] T. Sood, S. Prakash, S. Sharma, A. Singh, and H. Choubey, "Intrusion Detection System in Wireless Sensor Network Using Conditional Generative Adversarial Network," Wireless Personal Communications, vol. 126, no. 1, pp. 911-931, 2022/09/01 2022, doi: 10.1007/s11277-022-09776-x.

[6] Z. Zhang, Y. Zhao, A. Canes, D. Steinberg, and O. Lyashevska, "Predictive analytics with gradient boosting in clinical medicine," (in eng), Ann Transl Med, vol. 7, no. 7, p. 152, Apr 2019, doi: 10.21037/atm.2019.03.29.

[7] M. Faris, M. N. Mahmud, M. F. Mohd Salleh, and B. Alsharaa, "A differential evolution-based algorithm with maturity extension for feature selection in intrusion detection system," Alexandria Engineering Journal, vol. 81, pp. 178-192, 2023/10/15/ 2023, doi: https://doi.org/10.1016/j.aej.2023.09.032.

[8] R. Kohavi and G. H. John, "Wrappers for feature subset selection," Artificial Intelligence, vol. 97, no. 1, pp. 273-324, 1997/12/01/ 1997, doi: https://doi.org/10.1016/S0004-3702(97)00043-X.

[9] I. Kononenko, "Semi-naive bayesian classifier," Berlin, Heidelberg, 1991: Springer Berlin Heidelberg, in Machine Learning — EWSL-91, pp. 206-219.

[10] S. Shyam Sundar, R. Bhuvaneswaran, and L. SaiRamesh, "Ensemble Based Optimal Feature Selection Algorithm for Efficient Intrusion Detection in Wireless Sensor Network," KSII Transactions on Internet and Information Systems (TIIS), vol. 18, no. 8, pp. 2214-2229, 2024, doi: 10.3837/tiis.2024.08.009.

[11] L. Abualigah et al., "Optimizing Intrusion Detection in Wireless Sensor Networks via the Improved Chameleon Swarm Algorithm for Feature Selection," IET Communications, vol. 19, no. 1, p. e70029, 2025, doi: https://doi.org/10.1049/cmu2.70029.

[12] P. Ananth, S. Saravanan, S. Chinnathambi, and S. Karthikeyan, "IoT intrusion detection in wireless sensor networks: A comparative analysis of machine learning algorithms for enhanced security," AIP Conference Proceedings, vol. 3279, no. 1, 2025, doi: 10.1063/5.0262752.

[13] M. Devi, P. Nandal, and H. Sehrawat, "A lightweight approach for intrusion detection in WSNs based on DCGAN," International Journal of Information Technology, vol. 17, no. 2, pp. 951-957, 2025/03/01 2025, doi: 10.1007/s41870-024-02347-2.

[14] D. Karunkuzhali, K. P. Arunachalam, R. Ramamoorthi, and R. K. Kadu, "Cyber-physical system for enhanced WSN-IoT security using spherical graph triple convolutional neural network with planet optimization algorithm," Progress in Engineering Science, vol. 2, no. 3, p. 100108, 2025/09/01/ 2025, doi: https://doi.org/10.1016/j.pes.2025.100108.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, 27 Nov.-1 Dec. 1995 1995, vol. 4, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[16] P. Rahmani, M. Arefi, S. M. S. S. Shojae, and A. Mirzaee, "IoT-RNNEI: An Internet of Things Attack Detection Model Leveraging Random Neural Network and Evolutionary Intelligence," IET Communications, vol. 19, no. 1, p. e70055, 2025, doi: https://doi.org/10.1049/cmu2.70055.

[17] R. Saravana Ram and A. Gopi Saminathan, "An Intrusion Detection System in WSN Using an Optimized Self-Attention-Based Progressive Generative Adversarial Network," IETE Journal of Research, vol. 71, no. 4, pp. 1176-1189, 2025/04/03 2025, doi: 10.1080/03772063.2025.2452339.

[18] I. Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," Journal of Sensors, vol. 2016, p. 4731953, 2016/09/29 2016, doi: 10.1155/2016/4731953.

[19] T. S. L. V. Ayyarao et al., "War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization," IEEE Access, vol. 10, pp. 25073-25105, 2022, doi: 10.1109/ACCESS.2022.3153493.

[20] M. Chahardoli, N. O. Eraghi, and S. Nazari, "Namib beetle optimization algorithm: A new meta-heuristic method for feature selection and dimension reduction," Concurrency and Computation: Practice and Experience, vol. 34, no. 1, p. e6524, 2022, doi: https://doi.org/10.1002/cpe.6524.

[21] M. Sriraghavendra, M. Elsadig, I. H. Jaghdam, S. Abdel-Khalek, B. Galeebathullah, and S. Alkhalaf, "Knowledge Improved Hybrid DNN–KAN Framework for Intrusion Detection in Wireless Sensor Networks," IEEE Access, vol. 13, pp. 127558-127569, 2025, doi: 10.1109/ACCESS.2025.3586783.

[22] I. T. Jolliffe, Principal Component Analysis, 2nd edition (Springer Series in Statistics). New York, NY: Springer, 2002.

[23] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A survey on intrusion detection system in IoT networks," Cyber Security and Applications, vol. 3, p. 100082, 2025/12/01/ 2025, doi: https://doi.org/10.1016/j.csa.2024.100082.