

# CNN-GRU based cyber-attack classification and detection with the CICIDS-2017 dataset using optimization algorithm for honey badger

**Katikam Mahesh, Kunjam Nageswara Rao**

Department of Computer Science and Systems Engineering, AUCE (A), Andhra University, Visakhapatnam, India

## Article Info

### Article history:

Received Jan 5, 2025

Revised Apr 4, 2025

Accepted Jul 2, 2025

### Keywords:

Convolutional neural network

Deep neural network

Feature selection

Gated recurrent unit

Honey badger optimization

Network security

## ABSTRACT

The sheer volume of data exchanged has grown through information and communications technology (ICT) swiftly growing importance since the attackers benefit from illegal access to network data and introduce possible dangers for data theft or alteration. It is considered a significant barrier to monitor the network traffic for cyber-attack detection and classification with alarm ring to inform to network administrator. With KDD-CUP99, conventional machine learning methods like deep neural network (DNN), a kind of artificial neural network (ANN), cannot detect and classify novel attacks types and lacks clarity regarding accuracy. The CICIDS 2017 dataset, which is improved in this study, serves as training data for the model and useful framework that combines a hybrid convolutional neural network (CNN) with the gated recurrent unit (GRU) technique. The primary aim of this effort is to classify different security attacks and classify cyberthreats with honey badger optimization algorithm (HBOA). To strengthen the performance criteria for various assault types, such as F1-score, recall, precision, and others, the HBOA is utilized to modify the model parameters high-level features ought to be extracted from the network data using the hybrid model assessed and verified by simulation studies. The detection and classification output from the CNN-GRU model, which detects different security threats with greater accuracy of 94%.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Katikam Mahesh

Department of Computer Science and Systems Engineering, AUCE (A), Andhra University

Visakhapatnam-530003, India

Email: [katikammahesh@gmail.com](mailto:katikammahesh@gmail.com)

## 1. INTRODUCTION

Real-time security threat identification and mitigation depend on threat detection and response systems. These solutions forecast and stop threats in a variety of domains, including networks, cloud, endpoints, email, and applications through the application of artificial neural networks (ANNs), the k closest neighbors' conventional techniques in machine learning (ML) and deep learning (DL), decision trees (DT), and ANN. However, these ML and DL tricks still face challenges in raising detection rates and are unable to detect new attacks like malware attacks, web assaults, and lack of specific accuracy for various attack classes balance records are another big concern. The ML approaches that are now in use only work with tiny datasets for achieving all these goals, the technique suggest optimizes the optimization algorithm for honey badger increase model's power and accuracy, which were meant to extract high-level structures. After the network data an optimization algorithm and used to optimize the hybrid model's asset is used in this excellent attack detection method to protect network systems against. Various types of attacks to improve the convolutional

neural network-gated recurrent unit (CNN-GRU) model's presentation and increase its usefulness for attack detection, an optimization algorithm for honey badger is used in this study having an accuracy of 94% the suggested hybrid model uses a feature extraction and selection process that combines "recursive feature elimination" (RFE).

## 2. LITERATURE WORK

The global environment that makes it possible for electronic information to be freely used globally is referred to as a cyber system or wireless network. It serves as an international center for free access to resources and information. Data transfer and information exchange are currently dominated [1], [2]. Network security has altered significantly in response to the increase in cyberthreats in an effort to reduce cybercrimes. Technologies, experts, and processes are all part of network security, which aims to prevent attackers from misusing information [3]. Generally, for reducing cyberthreats: automated and traditional. While the traits of cyberattacks are constantly evolving, conventional approaches have drawbacks like imprecise detection, shoddy system settings, and limited data access. Kumar *et al.* [4], Dixit and Silakari [5] automated techniques that are able to identify emerging cybercrimes by learning from experience are the way of the future for cybersecurity. Cyber risks are a range of malicious actions aimed at collecting data, affecting data integrity, or causing harm to networks and computing devices [6]. Several illegal attacks, denial of service (DoS) harm, and other potential hazards belong to the potential attacks [7]-[9]. These attacks are able to steal key data and completely stop network operation. Discovering vulnerabilities and using several signature-based and rule-based strategies can help reduce [10], [11]. Because of their flexibility, security, and remarkable accuracy, several researchers have suggested applying ML and DL algorithms to secure network systems [12]-[14]. Both of these ideas have grown into fundamental strategies for preventing cyberthreats and overcoming the limitations of conventional defense systems. ML approaches have drawbacks despite their importance. These artificial intelligence subtype models, which are based on experience rather than explicit programming, may find use in a number of domains, including cybersecurity. In network systems, these methods are essential for detecting malware and malicious network behavior [15]. Conventional attack prevention techniques identify and categorize network attacks using preset standards and processes. They fall under a specific category of detecting skills and frequently struggle to identify novel forms of assault. Instead of following established rules and requirements, ML and DL algorithms can assess all network traffic [16]. Chesney *et al.* [17] and Yang *et al.* [18] developed algorithms that can accurately detect and forecast network attacks and monitor network traffic. This study intends to use a hybrid DL model that combines two approaches in order to produce effective outcomes for improving security of cyber security systems has been recognized in numerous past studies [19]-[21]. Assessing the usage of various machine learning techniques for network system security is the main aim of these review articles. The researchers who want to do their study in this field benefit greatly from these review analyses. However, the study of problems pertaining to cyber security in network systems has received little attention in these review articles, which have mostly focused on the application of ML techniques [22]. Three different methods were the subject of the study: support vector machines (SVM), DT, and deep belief networks (DBN). It demonstrates how urgently a new benchmark dataset is needed to determine latest developments in lack of diversity, missing data, and inadequate detection of advanced attacks are some of the present files' drawbacks. Custom learning models tailored to cybersecurity applications are in high demand. According to the review, further learning methods should be explored in order to detect cyberthreats. The study used a DL-based approach to identify various security threats as reported by Chethana *et al.* [23]. Black hole nodes in the path reduce network performance due to packet loss. The routing approach in this work reduces black hole node packet loss. Worm hole permeation into wireless networks may also result in improper routing and network segmentation. Regardless of the network density of wireless ad hoc networks, wormholes can be identified by a fast method that uses round trip duration and ordinal multidimensional scaling. Wormholes connected by both short and long paths can be identified by this method of detection. In order to ensure the detection of every possible wormhole and stop unnoticed assaults into the ad hoc network, experimental testing is carried out. DL models were used by Mughaid *et al.* [24] to build and execute a phishing detection method. The purpose of the DL model is to differentiate between phishing and anti-phishing communications by gathering both the standard features of email content and other data. The model is trained using three techniques. several sets of data. A comparison analysis demonstrated that the most accurate and practical findings were obtained when more features were used. The enhanced decision tree method in particular showed off its improved performance by successively achieving accuracy scores of 88%, 100%, and 97% across all datasets used. A hybrid CNN-GRU methodology for developing a network security described by Imrana *et al.* [25].

### 3. PROPOSED METHOD

The method of assault detection's design is founded on the security system's principle, which is to continually monitor network system data flow in order to detect and categorize various kinds of security assaults. A hybrid CNN and GRU system works like this to detect attacks to identify cyberattacks that are introduced into network systems by the attacker's malicious network assaults enable attackers to take advantage of system facts and gain unwanted gaining entry to a private network data. The suggested attack detection framework's initial goal is to create an effective model for Attribute selection with the help of metaheuristic HBOA algorithm.

#### 3.1. Data collection

The experimental analysis data used in this work originated from the CIC-IDS-2017 ("Canadian Institute for Cybersecurity Intrusion Detection Systems Evaluation Dataset 2017") dataset the collection ought to contain both attack data and network traffic. Each unique CSV file which holds the data relates to a different week's day or type of network activity. A total of 154,290 samples had been collected and taken to be used for the analysis, with an 80:20 split ratio across the training and validation data (123432 and 30858, respectively). Network traffic data which captures a range of activities and possibly hazardous habits, is captured for a certain day or time period and saved in a CSV file after uploading the datasets for each specific day the week's number of collected data samples have been gathered into one holistic dataset the idea is to output a single, sizable DataFrame by bringing together the several data-frames that reflect each day by combining network traffic data from the whole week into a single dataset, this produces a more carry out view of possible threats and network activity.

##### 3.1.1. The CIC-IDS-2017 dataset features

An accepted benchmark for training and analyzing intrusion detection systems is millions of examples of network flow data from real-world scenarios including different kinds of cyberthreats and attacks are presented. Instances in the labelled dataset can be classified as either malicious or benign traffic as shown in Figure 1.

```
# basic information about the dataset
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2830743 entries, 0 to 2830742
Data columns (total 79 columns):
 #   Column                                Dtype
---  -
 0   Destination Port                      int64
 1   Flow Duration                         int64
 2   Total Fwd Packets                     int64
 3   Total Backward Packets                int64
 4   Total Length of Fwd Packets           int64
 5   Total Length of Bwd Packets           int64
 6   Fwd Packet Length Max                 int64
 7   Fwd Packet Length Min                 int64
 8   Fwd Packet Length Mean                 float64
 9   Fwd Packet Length Std                  float64
10  Bwd Packet Length Max                 int64
11  Bwd Packet Length Min                 int64
12  Bwd Packet Length Mean                 float64
13  Bwd Packet Length Std                  float64
14  Flow Bytes/s                           float64
15  Flow Packets/s                         float64
16  Flow IAT Mean                          float64
17  Flow IAT Std                           float64
18  Flow IAT Max                           int64
19  Flow IAT Min                           int64
20  Fwd IAT Total                          int64
21  Fwd IAT Mean                          float64
22  Fwd IAT Std                            float64
23  Fwd IAT Max                            int64
24  Fwd IAT Min                            int64
25  Bwd IAT Total                          int64
26  Bwd IAT Mean                          float64
27  Bwd IAT Std                            float64
28  Bwd IAT Max                            int64
29  Bwd IAT Min                            int64
30  Fwd PSH Flags                          int64
31  Bwd PSH Flags                          int64
32  Fwd URG Flags                          int64
33  Bwd URG Flags                          int64
34  Fwd Header Length                     int64
35  Bwd Header Length                     int64
36  Fwd Packets/s                          float64
37  Bwd Packets/s                          float64
38  Min Packet Length                     int64
39  Max Packet Length                     int64
40  Packet Length Mean                     float64
41  Packet Length Std                      float64
42  Packet Length Variance                 float64
43  FIN Flag Count                         int64
44  SYN Flag Count                         int64
45  RST Flag Count                         int64
46  PSH Flag Count                         int64
47  ACK Flag Count                         int64
48  URG Flag Count                         int64
49  CWE Flag Count                         int64
50  ECE Flag Count                         int64
51  Down/Up Ratio                          int64
52  Average Packet Size                    float64
53  Avg Fwd Segment Size                   float64
54  Avg Bwd Segment Size                   float64
55  Fwd Header Length.1                     int64
56  Fwd Avg Bytes/Bulk                      int64
57  Fwd Avg Bulk Rate                       int64
58  Bwd Avg Bytes/Bulk                      int64
59  Bwd Avg Bulk Rate                       int64
60  Bwd Avg Bulk Rate                       int64
61  Subflow Fwd Packets                     int64
62  Subflow Bwd Packets                     int64
63  Subflow Fwd Bytes                       int64
64  Subflow Bwd Bytes                       int64
65  Init_Win_bytes_forward                  int64
66  Init_Win_bytes_backward                 int64
67  act_data_pkt_fwd                        int64
68  min_seg_size_forward                    int64
69  Active Mean                             float64
70  Active Std                              float64
71  Active Max                             int64
72  Active Min                             int64
73  Idle Mean                              float64
74  Idle Std                               float64
75  Idle Max                               int64
76  Idle Min                               int64
77  Label                                  object
dtypes: float64(24), int64(54), object(1)
memory usage: 1.7+ GB
```

Figure 1. CIC-IDS-2017 dataset features

### 3.2. Data preprocessing

Once a unified dataset has been created it is pre-processed to remove redundancies and uncertainties to guarantee data consistency various uncertainties are filtered out such as addressing missing values and eliminating duplicates. To guarantee preprocessing is necessary for guaranteeing the dataset's quality and integrity preceding additional analysis or modelling the EDA [exploratory data analysis] facilitates well-informed preprocessing decisions by offering insights into data characteristics furthermore, by establishing the values' upper and lower limits that constitute normal data points, the data's outliers are eliminated. Values that fall outside of these ranges are regarded as possible outliers. Values which vary in the upper and lower boundaries are substituted with the appropriate bounds for each column using the "NumPy's np" function by oversampling the synthetic minority oversampling technique (SMOTE) corrects unbalanced data as shown in Figure 2.

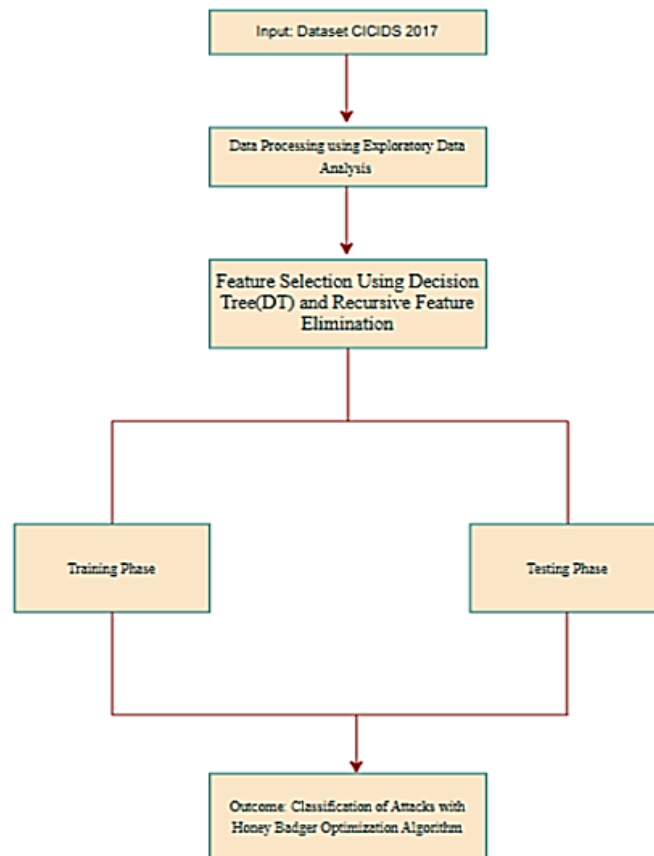


Figure 2. Framework for attack classification

#### 3.2.1. Handling the class imbalance using smote

The SMOTE method optimizes attack classification and detection to corrects unbalanced data and maintains the original training dataset's balance this technique generates synthetic instances rather than reorganizing minority class datasets this technique reduces overfitting problems that are frequently brought on by random oversampling. SMOTE creates new samples by interpolating across clustered positive samples while focusing on the feature space as shown in Figure 3.

### 3.3. Feature selection

The main and relevant characteristics are selected by employing a DTC technique and recursive feature elimination one feature selection technique for locating the dataset's salient features is the RFE. Once the least important features are continually eliminated until the required number of features is achieved a model that includes the remaining features is produced at first, the DTC is created and used as an estimate in this study 30 features with a step size of 1 were chosen for analysis making use of a recursive feature elimination.

```

from imblearn.over_sampling import SMOTE
sm = SMOTE()

# before applying the smote
y.value_counts()
3    10286
0    10000
4    10000
2    10000
10   10000
7     5933
6     5385
5     5228
11    3219
1     1953
12    1470
14     652
9       36
13       21
8        11
Name: Label, dtype: int64

X, Y = sm.fit_resample(x, y)

# after applying the smote
Y.value_counts()
0    10286
4    10286
2    10286
10   10286
3    10286
7    10286
6    10286
5    10286
11   10286
1    10286
12   10286
14   10286
9    10286
13   10286
8    10286
Name: Label, dtype: int64

```

Figure 3. SMOTE handling unbalanced data

### 3.4. Attack detection model

The capacity of CNN a popular neural network model that is frequently used in classification applications having three main stages such as layers that are completely connected (FC), pooling, and convolutional constitute the CNN architecture used in this study. The data on the attack is provided as input at the input layer CNNs' basic building blocks, known as convolutional layers use a number learnable filters or kernels that extract features by carrying over input images with data growing more abstract in deeper layers these filters aid in identifying particular patterns or characteristics such as the sort of attack as more layers are added the network's depth increases and its capacity to recognize intricate patterns is improved convolutional layers are followed by FC layers, which flatten one-dimensional vectors develop by relating the final feature as shown in Figure 4.

#### 3.4.1. CNN module

A CNN is composed of tens or hundreds of layers, each of which learns to identify distinctive elements of an image. The output of each convolved image is used as the input for the next layer, and each training image is made visible using various filter resolutions. The filters can start with very basic attributes like brightness and edges and work their way up to more detailed ones that give the object a distinct identity as shown in Figure 5.

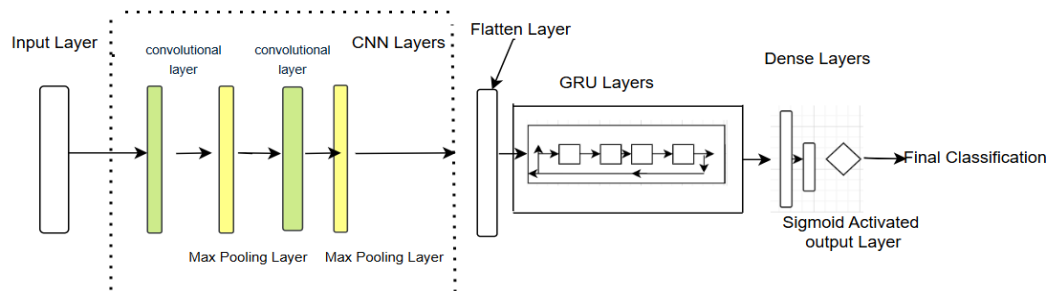


Figure 4. CNN-GRU model for attack detection

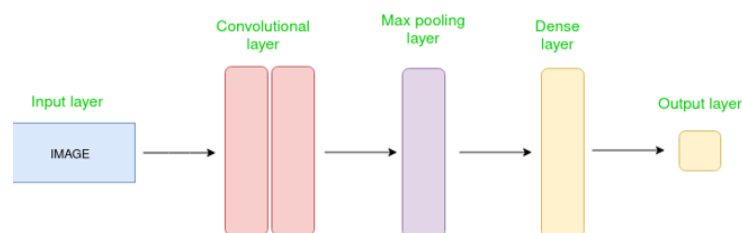


Figure 5. Architecture of a CNN module

Each potential class's class probabilities are produced by this layer, and the input image's projected class is the one with the highest probability. Or a data sample  $X$  denoted as that,  $X = \{x_1, x_2, x_n\}$ , a metric for weight The convolution operation in layer  $k$  is provided by  $W$ , a divergence of  $\delta$  given as shown in (1).

$$X_k = F(W_k * X_{k-1} + \delta_k) \quad (1)$$

where  $X_k$  is the  $k$ th kernel output,  $X_{k-1}$  is the input to the  $k$ th convolutional layer,  $*$  is the convolutional layer's operation, and  $F$  is the activation function ReLU. Following the convolution process, an activation function is applied to amplify differences among features extracted through convolution.

### 3.4.2. GRU module

A popular recurrent neural network type for data classification and prediction is the GRU. In this study, GRU allows the neuron to train the attack detection model and adaptively capture the dependencies on various time scales. GRU creates a memory mechanism by modularizing the neuron's internal data flow. By improving the ability to predict and classify data in attack detection, GRU raises the accuracy of detection.

The GRU architecture incorporates two gates one of which is a reset gate responsible for regulating the integration of new input and adjusting the existing memory state with the incoming input as shown in (2) and (3).

$$= \sigma(Wz * xt + Uz * ht - 1 + bz) \quad (2)$$

$$r = \sigma(Wr * xt + Ur * ht - 1 + br) \quad (3)$$

The update gate in the GRU is responsible for controlling and storing previous memory state and the hidden state as defined in (4) and (5).

$$\hat{h} = \tanh(Wh * xt + r * Uh * ht - 1 + bz) \quad (4)$$

$$h = z * ht - 1 + (1 - z) * \hat{h} \quad (5)$$

In this work, the GRU module is employed to extract temporal input data features through a recurrent process. Similar to the convolutional layer, GRU utilizes an activation function, utilizing functions such as tanh and hard sigmoid for implementation. The parameters of the CNN-GRU are optimized using a HBOA algorithm, which is discussed in the following section. GRU processes sequential data one element at a time, updating its hidden state according to the previous hidden state and the current input as shown in Figure 6.

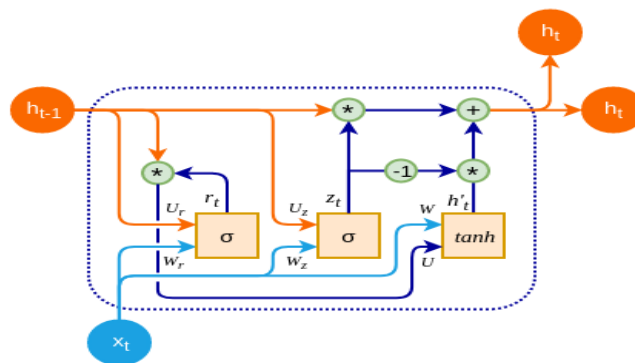


Figure 6. Architecture of a GRU cell

### 3.5. Optimization algorithm for honey badger

The pseudocode for the HBOA algorithm uses the CICIDS2017 dataset as input.

Step 1: at various points, scatter an expected number of honey badgers around the search area.

Step 2: measure each honey badger's level of fitness within the colony.

Step 3: give the best honey badger the maximum level of fitness.

Step 4: continue until each honey badger in the population satisfies a termination criterion. (e.g., the maximum number of iterations reached).

$$Z_i = LB_i + r_1 * (UB_i - LB_i), i = 1, 2, \dots, N, \quad (6)$$

where LB and UB represent the lower and upper bounds of the search space, respectively, and  $r_1$  is the random integer. The density factor ( $\alpha$ ) balances exploration (digging) with exploitation (honey), which is shown in (7).

$$\alpha = C * \exp(-t/T) \quad (7)$$

where C is the constant, which has a value, T is the total number of iterations, and t is the current iteration larger greater than 1. Furthermore, the solutions are updated using the Digging operators as shown in the (8).

$$Z_{new} = Z_b + F * \beta * I * Z_b + F * \alpha * di * r_3 * |\cos(2\pi r_4) * [1 - \cos(2\pi r_5)]| \quad (8)$$

$$F = \begin{cases} 1, & \text{if } (r_6 \leq 0.5), \\ -1 & \text{if Else} \end{cases} \quad (9)$$

Where, I is the smell intensity of the prey ( $x_b$ ) which is used for representing the distance between the  $x_b$  and  $x_i$  as shown in the (10) and (11).

$$I_i = r_2 * (S/4\pi d^2 i) \quad (10)$$

$$S = (Z_i - Z_i + 1)^2 \quad (11)$$

The solutions are updated using operators of honey stage and this process is achieved using the following formula;

$$Z_{new} = Z_b + F * r_1 * \alpha * di \quad (12)$$

where  $r_1$  is a random number. The steps of HBO are given as follows:

In the provided code the HBOA is used to optimize the hyper parameters of a CNN-GRU model the hyper parameters optimized by HBOA include the number of neurons in the CNN and GRU layers, kernel size, and pool size the objective is to find the combination of hyper parameters that maximizes the performance of the CNN-GRU model on the given dataset. The HBOA is implemented using the “mealy” library, which provides implementations of various nature-inspired optimization algorithms, including the honey badger algorithm. The problem to be optimized is defined as a dictionary containing the fitness function, lower and upper bounds for each hyper parameter optimization direction (minimize or maximize) and other optional parameters. The fitness function serves as the fitness function, which evaluates for a given set of hyper parameters the HBOA algorithm is instantiated with parameters such as the number of epochs, population size, and probability of random movement. The solve method of the HBOA algorithm is called to optimize the problem, and the best solution and fitness value are obtained this model is built and trained using the specified hyper parameters. After optimizing the model, the best solution (optimized hyper parameters) and corresponding fitness value are obtained.

#### 4. RESULTS AND DISCUSSION

There is much necessary to evaluate the model using metrics in terms of following:

Accuracy: displays the number of effectively identified cyberattacks derived by:

$$Accuracy = TP + TNTP + TN + FP + FN$$

Recall: a function's recall has been determined as the amount of correctly determined harms and is provided as

$$Recall = TPTP + FN$$

F1 rating: F1, which displays values ranging from 1 to 0 where 0 represents the lowest value and 1 represents the highest. Consequently, the following is the process for the F1 score:

$$F1\ rating = \frac{2 * Precision * Recall}{Precision + Recall}$$

Precision: Evaluates how accurate positive predictions out of all the positive predictions the model made.

$$Precision = \frac{TP + FP}{TP}$$

The term "true positive," or "TP," refers to the amount of correctly classified cyberattacks. The true negative, or TN, is a measure of the quantity of correctly identified normal attacks. The number of false cyberattack detections is known as the false positive score (FP), while the number of false cyberattack identifications is known as the false negative score (FN).

#### 4.1. Experimental results

The component of a dataset that you wish to gain an improved comprehension of is known as the target variable. With the help of the rest dataset, the user would want to predict this variable. The specific value or category that a model is attempting to predict or divide based on input features is known as the target data in machine learning (also known as the target variable, label, or output variable). It stands for the desired result that the model has been taught to discover from the input data as shown in Figure 7.

# Target column distribution

Data "Visualization:

"Label" df['Label'].value\_counts() df['Label'].plot(kind='barh') in.value\_counts() plt.xlabel('Count')

plt.ylabel('Attack Types')

plt.title('Target class distribution')

plt.show()

```
# Target column distribution
df['Label'].value_counts()
```

```
BENIGN                2096134
DoS Hulk              172846
DDoS                  128016
PortScan              90819
DoS GoldenEye         10286
FTP-Patator           5933
DoS slowloris         5385
DoS Slowhttptest      5228
SSH-Patator           3219
Bot                   1953
Web Attack - Brute Force 1470
Web Attack - XSS       652
Infiltration           36
Web Attack - Sql Injection 21
Heartbleed             11
Name: Label, dtype: int64
```

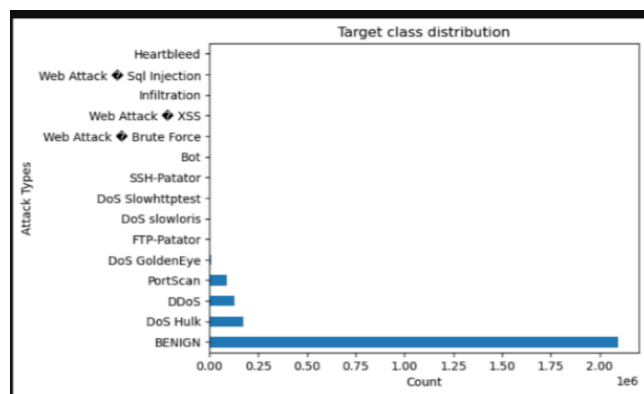


Figure 7. Depicts target class distribution

#### 4.2. Confusion matrix

The confusion matrix, displayed graphically as a table, is a method for evaluating a model's performance. It provides data practitioners with a more comprehensive insight of the model's performance, errors, and limitations. As seen in Figure 8, this allows data practitioners to refine their model and carry out further analysis. The results in Figure 8 tells us to avoding confusion between difference between true and false prediction for model.

conf\_matrix = confusion matrix (y\_test, predicted label)

print(conf\_matrix)

#### 4.3. Performance metrics

As per table, the CNN-GRU model's ideal accuracy was 94%. Furthermore, the CNN-GRU model's performance is contrasted with that of other classifiers, that include "K-nearest neighbor (KNN)", DT, and "artificial neural network (ANN)". The outcomes are demonstrated the proposed CNN-GRU model with honey badger algorithms which optimized CNN-GRU model shown in Table 1.



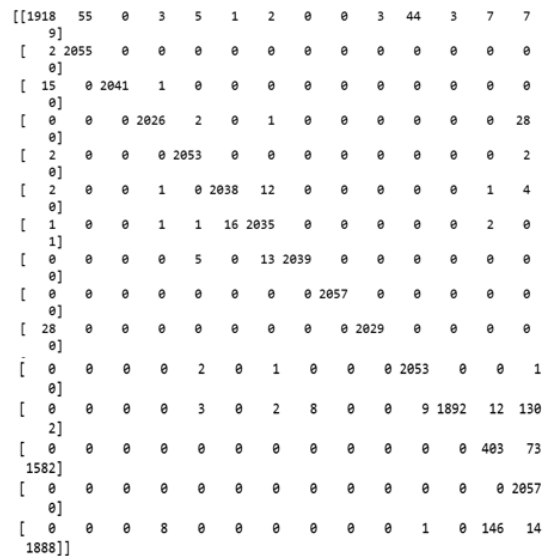


Figure 8. Confusion matrix

Table 1. Comparative analysis

Model	Metrics			
	Accuracy	Precision	Recall	F1 score
<b>KNN</b>	92%	92%	92%	92%
<b>DT</b>	91%	91%	91%	91%
<b>ANN</b>	91%	93%	91%	89%
<b>CNN-GRU-with honey badger algorithm</b>	94%	94%	94%	94%

## 5. CONCLUSION

To be able to recognize cyberthreats in the network system this research study explores the software of a hybrid classifier that connects CNN and GRU the introduction of possible attacks that impact the everyday operations of system networks is among the most important network security issues. A CNN-GRU model has been developed and simulated to accept and halt violence including Web, DoS, DDoS, SQL injection, and penetration threats for continuous network monitoring an HBOA algorithm is used to optimize the hybrid CNN-GRU system presented in the current paper the proposed method was tested on the CICIDS-2017 dataset with a split ratio of 80% for instructions and 20% for testing. This data had been pre-processed to guarantee consistency, and the SMOTE technique was applied to address the data imbalance problem an RFE with DT classifier is employed to choose and extract the dataset's most important and pertinent features in order to speed up the classification process. The CNN-GRU model is trained to classify data occurrences as either normal or abnormal. Based on the outcomes of multiple criteria employed to determine the hybrid model's performance, and this outperforms other ML models, attaining the highest accuracy of 94%. To achieve explain ability and interpretability in later work the project will explore the usage of generative AI models in addition to explainable AI models.

## ACKNOWLEDGEMENTS

For his guidance and support in reviewing and providing comments, we would like to sincerely thank Dr. Kunjam Nageswara Rao, Professor in the Department of Computer Science and System Engineering at Andhra University Andhra Pradesh, Visakhapatnam, India.

## FUNDING INFORMATION

Authors state no funding involved.

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Katikam Mahesh	✓	✓	✓	✓	✓	✓		✓	✓		✓			
Kunjam Nageswara Rao					✓					✓		✓	✓	

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review &amp; Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

## CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

## DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.





## REFERENCES

- [1] F. Ullah *et al.*, "Cyber security threats detection in internet of things using deep learning approach," *IEEE Access*, vol. 7, pp. 124379–124389, 2019, doi: 10.1109/ACCESS.2019.2937347.
- [2] J. Lee, J. Kim, I. Kim, and K. Han, "Cyber threat detection based on artificial neural networks using event profiles," *IEEE Access*, vol. 7, pp. 165607–165626, 2019, doi: 10.1109/ACCESS.2019.2953095.
- [3] Y. E. Sagduyu, Y. Shi, and T. Erpek, "IoT network security from the perspective of adversarial deep learning," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2019, vol. 2019-June, pp. 1–9, doi: 10.1109/SAHCN.2019.8824956.
- [4] C. Kumar, T. S. Bharati, and S. Prakash, "Online social network security: a comparative review using machine learning and deep learning," *Neural Processing Letters*, vol. 53, no. 1, pp. 843–861, 2021, doi: 10.1007/s11063-020-10416-3.
- [5] P. Dixit and S. Silakari, "Deep learning algorithms for cybersecurity applications: a technological and status review," *Computer Science Review*, vol. 39, p. 100317, Feb. 2021, doi: 10.1016/j.cosrev.2020.100317.
- [6] I. H. Sarker, "Deep Cybersecurity: A comprehensive overview from neural network and deep learning perspective," *SN Computer Science*, vol. 2, no. 3, p. 154, May 2021, doi: 10.1007/s42979-021-00535-6.
- [7] J. M. Biju, N. Gopal, and A. J. Prakash, "Cyber-attacks and its different types," *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, no. 3, pp. 4849–4852, 2019.
- [8] S. Zaman *et al.*, "Security threats and artificial intelligence based countermeasures for internet of things networks: a comprehensive survey," *IEEE Access*, vol. 9, pp. 94668–94690, 2021, doi: 10.1109/ACCESS.2021.3089681.
- [9] N. Mishra and S. Pandya, "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: a systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021, doi: 10.1109/ACCESS.2021.3073408.
- [10] Z. S. Malek, B. Trivedi, and A. Shah, "User behavior pattern -signature based intrusion detection," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, Jul. 2020, pp. 549–552, doi: 10.1109/WorldS450073.2020.9210368.
- [11] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Rule generation for signature-based detection systems of cyber-attacks in iot environments," *Bulletin of Networking, Computing, Systems, and Software*, vol. 8, no. 2, pp. 93–97, 2019.
- [12] C. Gupta, I. Johri, K. Srinivasan, Y. C. Hu, S. M. Qaisar, and K. Y. Huang, "A systematic review on machine learning and deep learning models for electronic information security in mobile networks," *Sensors*, vol. 22, no. 5, 2022, doi: 10.3390/s22052017.
- [13] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecurity: a review," *IEEE Access*, vol. 10, no. 10, pp. 19572–19585, 2022, doi: 10.1109/ACCESS.2022.3151248.
- [14] R. Geetha and T. Thilagam, "A review on the effectiveness of machine learning and deep learning algorithms for cyber security," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2861–2879, Jun. 2021, doi: 10.1007/s11831-020-09478-2.
- [15] Y.-H. Choi *et al.*, "Using deep learning to solve computer security challenges: a survey," *Cybersecurity*, vol. 3, no. 1, p. 15, Dec. 2020, doi: 10.1186/s42400-020-00055-5.
- [16] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2021, doi: 10.1007/s11831-020-09496-0.
- [17] S. Chesney, K. Roy, and S. Khorsandroo, "Machine learning algorithms for preventing IoT cybersecurity attacks," *Advances in Intelligent Systems and Computing*, vol. 1252 AISC, pp. 679–686, 2021, doi: 10.1007/978-3-030-55190-2\_53.
- [18] H. Yang, R. Zeng, G. Xu, and L. Zhang, "A network security situation assessment method based on adversarial deep learning," *Applied Soft Computing*, vol. 102, p. 107096, Apr. 2021, doi: 10.1016/j.asoc.2021.107096.
- [19] K. R. Dalal and M. Rele, "Cyber security: threat detection model based on machine learning algorithm," in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, Oct. 2018, pp. 239–243, doi: 10.1109/CESYS.2018.8724096.
- [20] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. Minhaz Hossain, S. Ikhlaiq, and S. Hossain, "Cyber intrusion detection using machine learning classification techniques," in *Communications in Computer and Information Science*, vol. 1235 CCIS, 2020, pp. 121–131.
- [21] M. A. Ferrag, L. Maglaras, S. Moschogiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.





- [22] K. Shaukat, S. Luo, S. Chen, and D. Liu, "Cyber threat detection using machine learning techniques: a performance evaluation perspective," in *2020 International Conference on Cyber Warfare and Security (ICWWS)*, Oct. 2020, pp. 1–6, doi: 10.1109/ICWWS48432.2020.9292388.
- [23] C. Chethana, P. K. Pareek, V. H. Costa de Albuquerque, A. Khanna, and D. Gupta, "Deep learning technique based intrusion detection in cyber-security networks," in *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, Oct. 2022, pp. 1–7, doi: 10.1109/MysuruCon55714.2022.9972350.
- [24] A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, "An intelligent cyber security phishing detection system using deep learning techniques," *Cluster Computing*, vol. 25, no. 6, pp. 3819–3828, Dec. 2022, doi: 10.1007/s10586-022-03604-4.
- [25] Y. Imrana, Y. Xiang, L. Ali, A. Noor, K. Sarpong, and M. A. Abdullah, "CNN-GRU-FF: a double-layer feature fusion-based network intrusion detection system using convolutional neural network and gated recurrent units," *Complex & Intelligent Systems*, vol. 10, no. 3, pp. 3353–3370, Jun. 2024, doi: 10.1007/s40747-023-01313-y.

## BIOGRAPHIES OF AUTHORS



**Katikam Mahesh**     currently as a research scholar at Andhra University College of Engineering in Visakhapatnam India in the Department of Computer Science and Systems Engineering. He pursued his Master's in computer science and engineering from SISTAM College in Srikakulam. His area of interest includes network security and machine learning. He has around 02 publications in the area of networks security and machine learning. He can be contacted at email: [katikammahesh@gmail.com](mailto:katikammahesh@gmail.com).



**Dr. Kunjam Nageswara Rao**     working as professor and head in the Department of Computer Science and Systems Engineering, University College of Engineering, Andhra University, Visakhapatnam, Andhra Pradesh, India. He completed his B. Tech in computer science and systems engineering from GITAM Engineering College, M. Tech in computer science and technology from Andhra University and Ph.D. in computer science engineering from Andhra University. His current research includes bio-informatics, computer science information systems, health informatics, big data analytics, artificial intelligence and machine learning, algorithm and design analysis. He can be contacted at email: [kunjamnag@gmail.com](mailto:kunjamnag@gmail.com).