World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Resilient IoT Security: Early Flood Attack Detection in IoT Networks Using GRU Deep Learning Model

Mildred Adwubi Bonsu * and Philip Akekudaga

College of Emergency Preparedness, Homeland Security and Cybersecurity, University at Albany, State University of New York. USA

World Journal of Advanced Research and Reviews, 2025, 27(02), 871-886

Publication history: Received on 28 June 2025; revised on 10 August 2025; accepted on 12 August 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.27.2.2897

Abstract

Securing Internet of Things (IoT) networks has become increasingly critical as their integration across essential sectors continues to expand. Among the most pressing threats are flood attacks, a form of Distributed Denial of Service (DDoS) that overwhelms network resources and causes service degradation. In this study, the detection of flood attacks in IoT environments is addressed using a deep learning model based on the Gated Recurrent Unit (GRU) architecture. Within the scope of the analysis, the CICIoT2023 dataset, which reflects realistic IoT traffic and attack behavior, was employed for training and validation. The results have shown that the flood attacks were successfully detected, and the model achieved an accuracy score of 0.98, with moderate precision, recall, and F1 scores. In this way, flood attacks in IoT can be identified early to mitigate their impact and enhance the resilience of IoT infrastructure. This study contributes to intelligent IoT security by integrating updated datasets, sequential modeling, and empirical evaluation, establishing a solid foundation for future research in threat detection systems.

Keywords: Internet Of Things (IoT); IoT Security; Distributed Denial of Service (DDOS); Deep Learning; Gated Recurrent Unit (GRU).

1. Introduction

The Internet of Things (IoT) has become an integral part of daily life, transforming how we manage our homes, communicate, and operate across various industries. With IoT devices playing a crucial role in sectors such as healthcare, transportation, energy, and smart homes, they offer unprecedented convenience and efficiency. However, as their presence grows, so do the security challenges associated with their widespread interconnectivity. Among the most critical threats to IoT networks are flood attacks, a form of Distributed Denial of Service (DDoS) attack that disrupts the normal functionality of devices and networks by overwhelming them with illegitimate traffic [1, 2]. Flood attacks targeting IoT infrastructure have escalated in recent years, presenting significant real-world implications. In 2022, the number of IoT malware attacks worldwide reached 112.29 million, marking an 87% year-over-year increase from 2021. By the fourth quarter of 2024, global DDoS attacks had risen to 512,000, up from 274,000 in the first quarter of 2023 [3]. These incidents often result in service disruptions, data loss, system downtime, and substantial reputational damage, particularly for organizations that rely on real-time data transmission.

Traditional security solutions, such as signature-based intrusion detection systems and basic firewalls, have proven inadequate in the face of these evolving attack patterns. Notably, many legacy systems are not optimized for IoT environments' dynamic, resource-constrained, and heterogeneous nature [4]. As a result, there is an urgent need for more adaptive and intelligent detection methods. This study aims to provide a solution to improve the efficiency of the detection of flood attacks in IoT environments with deep learning using the Gated Recurrent Unit (GRU) algorithm, which is effective in capturing temporal dependencies in sequential data. The model is trained and evaluated using the

* Corresponding author: Mildred Adwubi Bonsu

CICIoT2023, a comprehensive dataset curated by the Canadian Institute for Cybersecurity [5], which includes a diverse set of simulated IoT traffic, encompassing both benign and malicious activities. The study has three research objectives: to improve the accuracy in detecting flood attacks in the IoT environment, develop an optimal deep learning model capable of detecting compromises in security within the IoT environment, and leverage the potential of deep learning to improve both the false positive and true positive rate metrics. The research adopts a rigorous experimental methodology grounded in deep learning principles, to assess the model's performance. Key performance metrics, such as accuracy, precision, recall, F1-score, and ROC-AUC, are used to evaluate the model's effectiveness in detecting flood attacks. This study addresses the following key research questions; 1. How can the efficiency of a deep learning-based model be improved in the detection of flood attacks in an IoT context? 2. What are the key parameters that are to be considered in developing a deep learning model to enhance its applicability in identifying security compromises in real-world IoT systems? 3. How can the proposed deep learning model be designed to improve its rate of true positives while maintaining a low rate of false alarms in flood attack detection? This study contributes to improving proactive threat detection systems in IoT environments. The findings are expected to provide valuable insights for the deployment of more robust security mechanisms in IoT systems, particularly those vulnerable to DDoS-related disruptions.

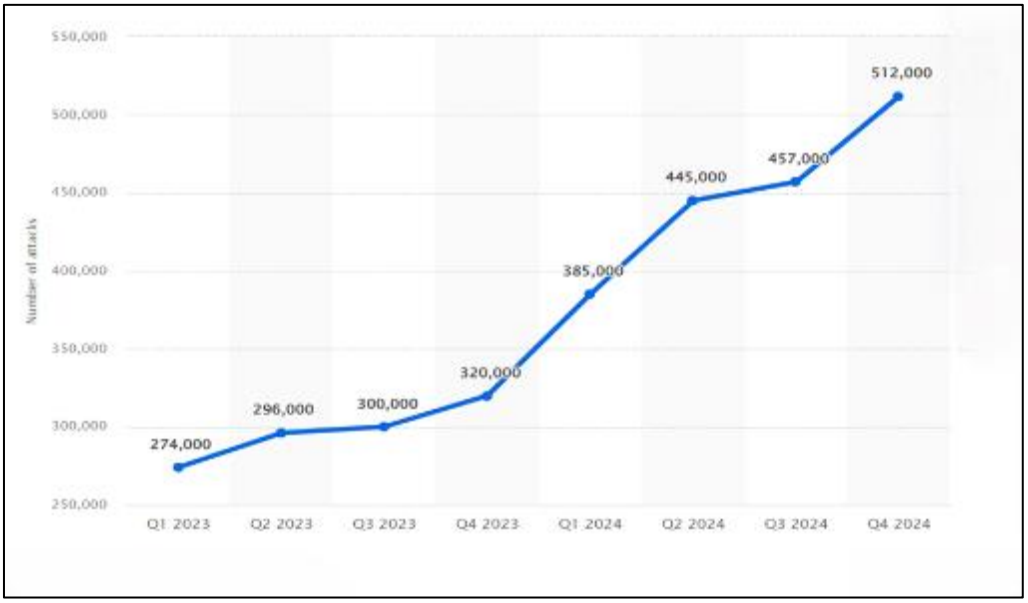


Figure 1 Number of DDoS Attacks Worldwide from 1st Quater 2023 to 4th Quater 2024 (Source: Statista)

Among the various flood attack and DDoS detection approaches developed in previous studies, several significant issues persist. Key challenges include the time required to identify attacks, detection accuracy, and the realism of the approach. These challenges often depend on the type of dataset and the features selected to represent the attack classes. A review of the literature reveals that many studies used outdated, small, or imbalanced datasets, which hindered the models' ability to effectively identify certain types of attacks. Additionally, some solutions sacrificed accuracy for speed of execution. These challenges are primarily due to the datasets used to train deep learning models. Training on a more current, real-time dataset could improve the model's ability to detect attacks in real-world scenarios. These limitations highlight the need for further investigation and the development of optimal solutions to enhance the efficiency of flood attack detection. Deep learning techniques have shown promising results, but several issues must be addressed. Many studies trained, tested, and validated their models on small datasets, which may not accurately reflect real-world conditions. Furthermore, some studies did not address the computational complexity of their models or provide adequate interpretation of their findings, which are critical for practical implementation. This study aims to address these challenges by improving both the false alarm rate and detection accuracy using a more recent and real-time dataset.

The contribution of this work is threefold. First, it contributes to the existing body of literature by offering a comparative assessment of deep learning techniques tailored to the detection of flood-based attacks in IoT networks. The study presents quantifiable performance metrics, including accuracy, recall, and F1-score, under realistic conditions, thereby offering a reference point for future experimental replication and optimization. Second, the study provides a model-driven perspective on integrating sequential learning into intrusion detection systems. By illustrating how GRU-based models can be tuned for pattern recognition in noisy and heterogeneous IoT traffic, the research advances the methodological foundation for low-overhead, high-accuracy detection in constrained network environments. Lastly,

this work yields practical value for the broader cybersecurity community. Network engineers, system architects, and regulatory stakeholders can utilize the findings to inform design choices in IoT network defense architectures, establish baseline detection capabilities, and align with emerging standards for secure device interoperability. The study thus contributes not only to academic inquiry but also to applied efforts aimed at strengthening the resilience of next-generation IoT infrastructures.

The study is organized into five sections. Section 1. introduces the study, providing an overview of the research problem and questions. It also presents the need for effective detection of flood attacks in IoT networks. Section 2. presents the background of the study, reviews the relevant literature, and summarizes existing related work. Section 3. presents the conceptual framework and methodology employed in this study, including the research design, data collection procedures, pre-processing strategies, model architecture, training and testing protocols, and performance evaluation criteria. Section 4. presents the results and discusses the findings, while Section 5. concludes the paper by discussing the study's limitations and directions for future research.

2. Materials And Methods

This section presents the methodology used in the study, including the research design, conceptual framework, data collection, preparation and pre-processing, cross-validation, suggested model creation, training and testing processes, performance evaluation, and methodological overview are all covered in detail. Figures 4 and 9 show Python code used for the experiments.

2.1. Research Design

This study's experimental research design entails the creation and assessment of a deep learning-based model. The purpose of the study is to develop a superior model based on a Gated Recurrent Unit (GRU) that enhances the rate at which flood attacks in IoT are detected.

2.2. Conceptual Framework

The conceptual framework encompasses the various stages of the research process. It includes data acquisition and description, data preparation and pre-processing, development of the proposed model, training and testing procedures, cross-validation, and performance metrics. These components form the foundation for the development and evaluation of the proposed optimized neural network model.

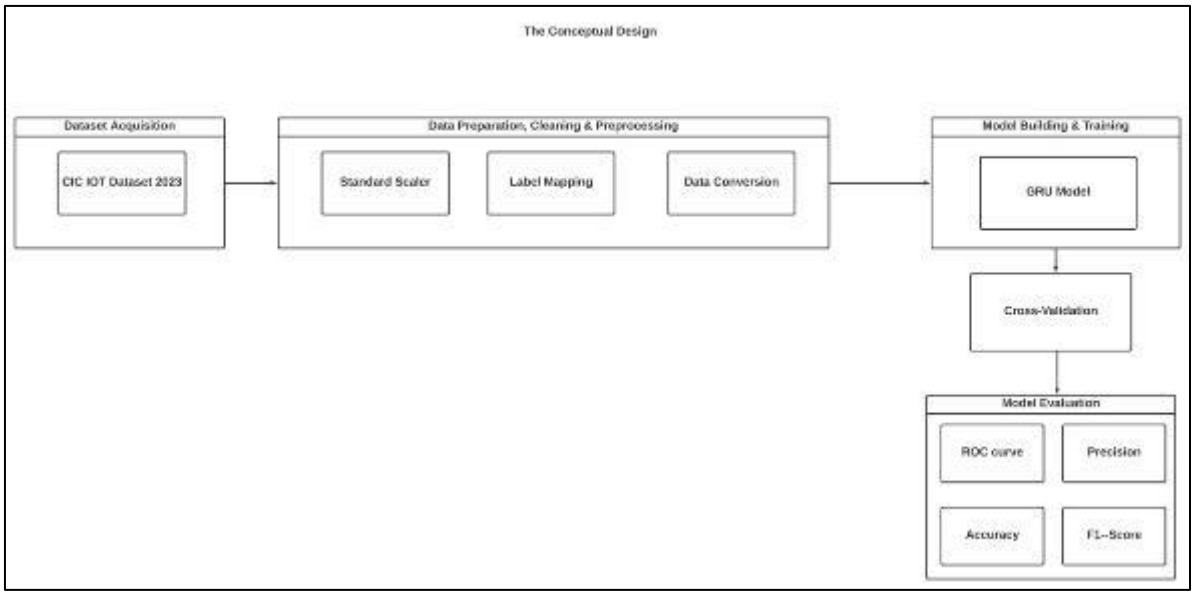
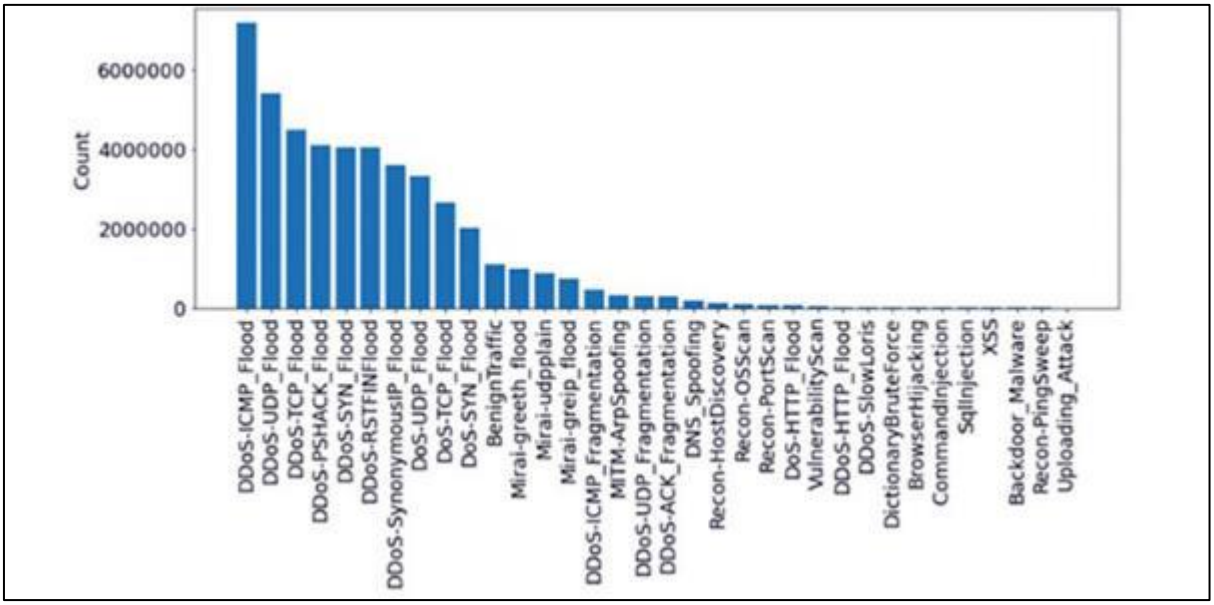


Figure 2 Conceptual Framework

2.3. Dataset Acquisition

The model proposed in this study was trained and tested using the CICIoT2023. This dataset was created to represent as closely as possible, real-world DDOS attack scenarios, especially those of flood attacks. It contains a balanced set of

seven categories of DDoS attacks in the IoT context. The CICIoT2023 is publicly available on the Canadian Institute for Cybersecurity website[5]. In Figure 3, a complete breakdown of the CICIoT2023 with the various types of classes is presented.



Source: Canadian Institute for Cybersecurity website

Figure 3 The complete dataset breakdown.

Neto et al. [5] set up several devices that imitate a real-world installation of IoT devices and services and configure traffic monitors on them to capture attack data. Each attack involves a unique experiment that involves all relevant devices. In the end, the count for each of the thirty-three categories of attack is illustrated as seen in Figure 3. It is clear from this graph that the authors gathered an extensive amount of flood attacks, making this dataset an ideal choice for training the deep learning model.

2.4. Data Preparation, Cleaning, and Pre-processing

Data pre-processing is essential for preparing raw data for deep learning models, especially when the data is incomplete or inconsistent. In this study, the dataset was first cleaned by removing irrelevant, redundant, or erroneous entries, handling missing values through removal, and discarding outliers or infinite values to ensure data integrity. Once cleaned, the data was transformed and normalized to fit the Gated Recurrent Unit (GRU) model's input requirements. Normalization standardizes the data, ensuring that all features are on a comparable scale, which helps improve model efficiency and accuracy. The pre-processing steps were executed using Python libraries such as Pandas and NumPy. Key tasks involved:

Standard Scaling: Each feature was scaled to have a mean of 0 and a standard deviation of 1, optimizing activation functions like sigmoid and tanh, which perform best with scaled inputs. This ensures improved model convergence.

Label Mapping: Categorical data in the CICIoT2023 was converted into numerical form by assigning a unique integer to each category. This transformation enabled the neural network to process the data and make predictions.

Data Conversion: The extracted features and labels were converted into NumPy arrays, making the dataset compatible with the proposed GRU model for training.

2.5. Model Building and Training

The proposed recurrent model is trained with the TensorFlow framework. The model was also trained and validated using K-fold cross-validation. After scaling and mapping the labels to be used, the model is developed. A sequential neural network is defined with a GRU layer for the model. This layer returns a linear sequence of data and has 64 units. After this layer, a batch normalization layer is added to normalize the output from the first layer, hence ensuring that the training process remains stable. A new GRU layer of 32 units that returns a single output with a batch normalization layer is then added to the model's architecture. A fully connected dense layer with a "softmax" activation function is

now applied since the model will classify multi-classes. Now, the k-fold validation process is initialized to five with the shuffle option set to true. The hyperparameters, which are the learning rate, batch size, and the number of epochs, for training the model are specified. The epoch specifies how many times the training process is to be iterated. A third dimension is added to change the shape of the model. The model is trained with the Keras framework. It has to do with compiling the model, defining its metrics, loss, and optimizer, and training the model with the training and validation data. In the compilation phase, the loss, optimizer, and metrics of the model are configured. The Sparse Categorical Cross-entropy is the typical loss function used in this study. For the optimizer, the Adam algorithm is used on the specified learning rate. This algorithm adjusts the learning rate during training. During evaluation and training, the accuracy metric will be determined and communicated. The model is now trained using the training and validation data (X_train, X_val) and its corresponding target labels (y_train, y_val). The performance of the model on the training and validation data is assessed after each epoch, keeping a record of the accuracy metric.

2.5.1. The Gated Recurrent Unit

Gated Recurrent Units (GRUs) are designed to capture model dependencies in sequential data.
With sequential data, every input depends on the ones before it. So, GRUs have a hidden state
(ht) that extracts information from the previous time step, for updating this state at each time
step. GRUs are composed mainly of two gates which are the Update (zt) and Reset gates (rt)
which decides how much of the past information is to be passed along and those that are to be forgotten, respectively. The new hidden state (ht) is a combination of the previous hidden state
(ht-1) and a candidate hidden state (~ht) whereas the ~ht is a weighted combination of the
previous hidden state (ht-1) and the current input (xt). These weights are given by the reset gate. The mathematical expressions of how the candidate hidden state (~ht), update gate (zt), and reset gate (rt) are calculated are shown below:

$$\tilde{h}_t = \tanh(W_{hx}x_t + r_t \odot (W_{hh}h_{t-1}))$$
$$z_t = \sigma(W_{hz}x_t + U_{hz}h_{t-1})$$
$$r_t = \sigma(W_{rx}x_t + U_{rh}h_{t-1})$$

In the expressions, the W and U are the weight matrices, (⊙) denotes element-wise multiplication and (σ) is the sigmoid activation function. The final hidden state (ht) is then derived from combining the ht-1 and the ~ht, which are weighted by the update gate.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

2.5.2. Cross-validation

This process is carried out to accurately estimate the performance of a deep learning model and its ability to be generalized. This is done to take care of overfitting. For this study, the K-fold cross-validation method was employed. The dataset is split into five subsets (K folds), and the model is repeatedly trained and assessed on portions of the subsets. To take care of bias, the data is first shuffled before it is partitioned into folds.

2.5.3. Model Evaluation

The trained model is now tested on the validation dataset. With the input features of the validation data (X_val) and the target labels of the validation data (y_val), the performance of the model is calculated, returning the validation loss and validation accuracy as its results. For this study, we accumulate and keep track of the different validation accuracy and validation losses at each iteration of the cross-validation.

2.6. Performance Metrics

The trained model is tested on the validation dataset, using the input features (X_val) and target labels (y_val) to calculate validation loss and accuracy. These metrics are tracked across iterations during cross-validation. The model's performance is assessed using accuracy, precision, F1-score, and the ROC curve. Key performance indicators include true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). TP refers to correctly predicted positive data points, TN to correctly predicted negative data points, FP to incorrect positive predictions, and FN to

incorrect negative predictions. Accuracy, precision, F1-score, and ROC curve are metrics that offer quantitative evaluations of how well the model can spot flood attacks.

- Accuracy score refers to the ratio of true predicted labels to the total number of labels. It measures how efficiently the model performs.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad \text{----------} \quad (1)$$

Source: G.Ahmed,[39]

- Precision focuses on the number of the model's predicted true positives that are really, true positives.

$$Precision = \frac{TP}{(TP + FP)} \quad \text{----------} \quad (2)$$

Source: G.Ahmed, [39]

- Receiver Operating Characteristic Curve (AUC) is a metric that gives a quantitative value of the overall classification performance at all thresholds by the model.

Source: Yousuf and Mir [35]

- Recall score provides a quantitative measure of the proportion of true positives predicted by the model against the actual positive cases.

$$Recall = \frac{TP}{(TP + FN)} \quad \text{----------} \quad (3)$$

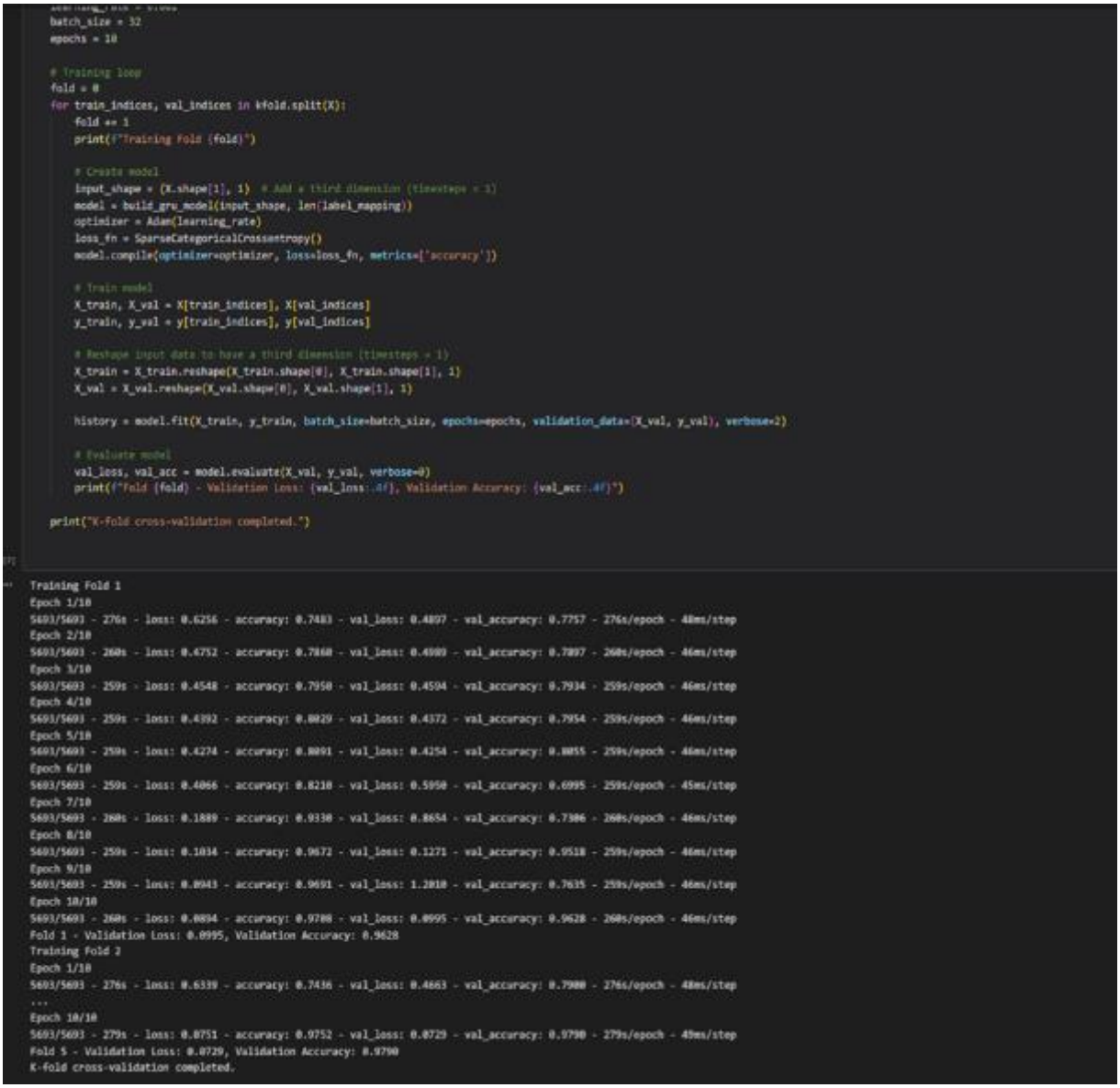
Source: G.Ahmed,[39]

- F1-score provides an evaluation of the performance of the model by calculating the mean between precision and recall.

$$F1\ Score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \quad \text{----------} \quad (4)$$

Source: Brownlee,[45]

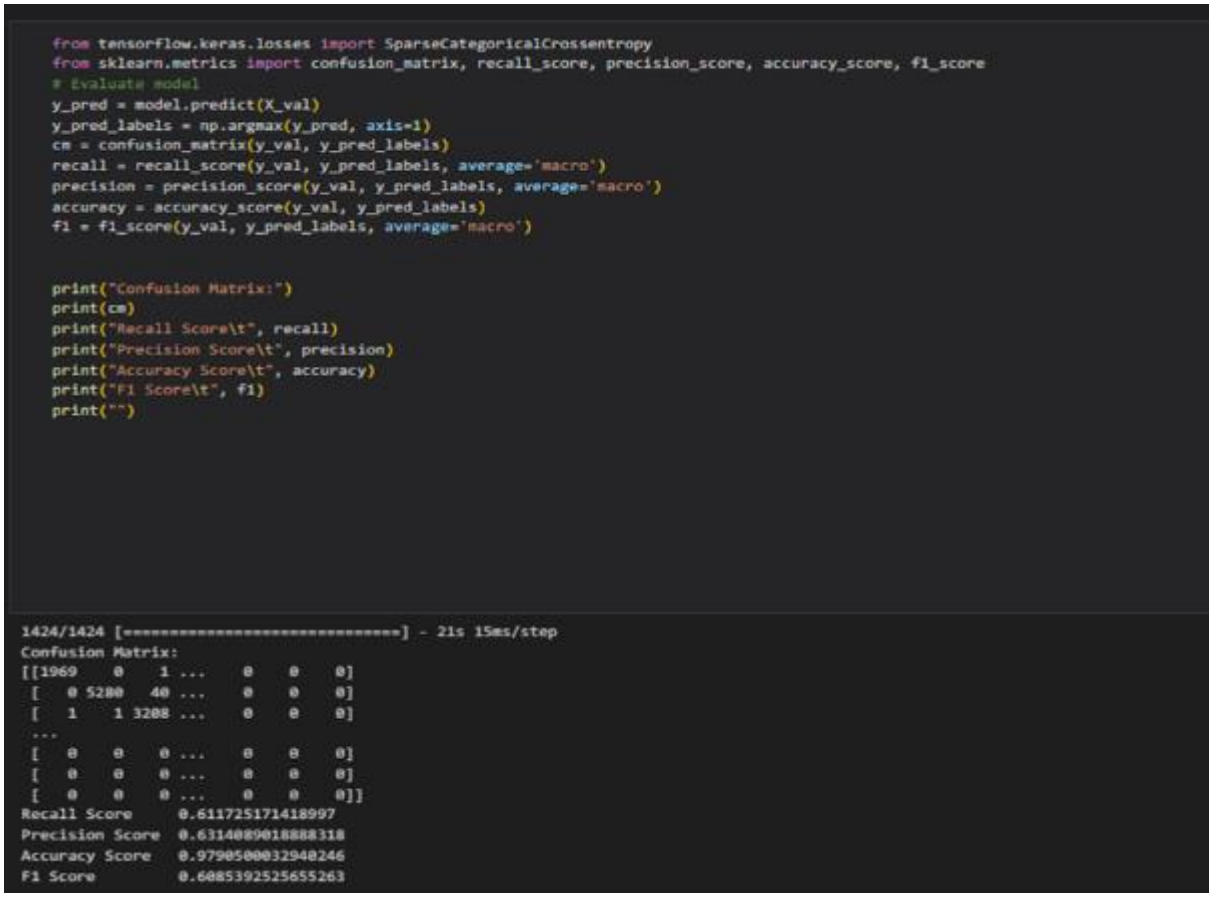
The accuracy (1) of the model is obtained by dividing the overall number of the model's correctly predicted cases (TP + TN) by the total number of predictions (TN + TP + FP + FN) it made. For the precision (2) of the model, the focus is set only on the ratio of correct positive predictions (TP) out of the total positive predictions (TP + FP) by the model. Regarding the recall metric (3), the total number of correctly predicted positive instances (TP) is divided by the sum of the number of correctly predicted positive instances and the number of wrongly predicted negative instances (TP + FN). Lastly, the F1-score (4) presents quantitative data on the balance between the model's precision (2) and recall (3). After several experiments, the performance of the model was measured based on accuracy, precision, recall, ROC, and F1-score.



```
batch_size = 32
epochs = 10

# Training loop
fold = 0
for train_indices, val_indices in kfold.split(X):
    fold += 1
    print(f"Training Fold {fold}")

    # Create model
    input_shape = (X.shape[1], 1)  # Add a third dimension (timesteps = 1)
    model = build_gru_model(input_shape, len(label_mapping))
    optimizer = Adam(learning_rate)
    loss_fn = SparseCategoricalCrossentropy()
    model.compile(optimizer=optimizer, loss=loss_fn, metrics=['accuracy'])

    # Train model
    X_train, X_val = X[train_indices], X[val_indices]
    y_train, y_val = y[train_indices], y[val_indices]

    # Reshape input data to have a third dimension (timesteps = 1)
    X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
    X_val = X_val.reshape(X_val.shape[0], X_val.shape[1], 1)

    history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(X_val, y_val), verbose=2)

    # Evaluate model
    val_loss, val_acc = model.evaluate(X_val, y_val, verbose=0)
    print(f"Fold {fold} - Validation Loss: {val_loss:.4f}, Validation Accuracy: {val_acc:.4f}")

print("K-Fold cross-validation completed.")
```

```
Training Fold 1
Epoch 1/10
5693/5693 - 276s - loss: 0.6256 - accuracy: 0.7483 - val_loss: 0.4897 - val_accuracy: 0.7757 - 276s/epoch - 48ms/step
Epoch 2/10
5693/5693 - 260s - loss: 0.4752 - accuracy: 0.7868 - val_loss: 0.4909 - val_accuracy: 0.7897 - 260s/epoch - 46ms/step
Epoch 3/10
5693/5693 - 259s - loss: 0.4548 - accuracy: 0.7950 - val_loss: 0.4594 - val_accuracy: 0.7934 - 259s/epoch - 46ms/step
Epoch 4/10
5693/5693 - 259s - loss: 0.4392 - accuracy: 0.8029 - val_loss: 0.4372 - val_accuracy: 0.7954 - 259s/epoch - 46ms/step
Epoch 5/10
5693/5693 - 259s - loss: 0.4274 - accuracy: 0.8091 - val_loss: 0.4254 - val_accuracy: 0.8055 - 259s/epoch - 46ms/step
Epoch 6/10
5693/5693 - 259s - loss: 0.4066 - accuracy: 0.8210 - val_loss: 0.5950 - val_accuracy: 0.6995 - 259s/epoch - 45ms/step
Epoch 7/10
5693/5693 - 260s - loss: 0.1889 - accuracy: 0.9330 - val_loss: 0.8654 - val_accuracy: 0.7386 - 260s/epoch - 46ms/step
Epoch 8/10
5693/5693 - 259s - loss: 0.1034 - accuracy: 0.9672 - val_loss: 0.1271 - val_accuracy: 0.9518 - 259s/epoch - 46ms/step
Epoch 9/10
5693/5693 - 259s - loss: 0.0943 - accuracy: 0.9691 - val_loss: 1.2810 - val_accuracy: 0.7635 - 259s/epoch - 46ms/step
Epoch 10/10
5693/5693 - 260s - loss: 0.0894 - accuracy: 0.9708 - val_loss: 0.0995 - val_accuracy: 0.9628 - 260s/epoch - 46ms/step
Fold 1 - Validation Loss: 0.0995, Validation Accuracy: 0.9628
Training Fold 2
Epoch 1/10
5693/5693 - 276s - loss: 0.6339 - accuracy: 0.7436 - val_loss: 0.4863 - val_accuracy: 0.7980 - 276s/epoch - 48ms/step
...
Epoch 10/10
5693/5693 - 279s - loss: 0.0751 - accuracy: 0.9752 - val_loss: 0.0729 - val_accuracy: 0.9790 - 279s/epoch - 49ms/step
Fold 5 - Validation Loss: 0.0729, Validation Accuracy: 0.9790
K-fold cross-validation completed.
```

Figure 4 Python Code showing model performance per epoch



```
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from sklearn.metrics import confusion_matrix, recall_score, precision_score, accuracy_score, f1_score
# Evaluate model
y_pred = model.predict(X_val)
y_pred_labels = np.argmax(y_pred, axis=1)
cm = confusion_matrix(y_val, y_pred_labels)
recall = recall_score(y_val, y_pred_labels, average='macro')
precision = precision_score(y_val, y_pred_labels, average='macro')
accuracy = accuracy_score(y_val, y_pred_labels)
f1 = f1_score(y_val, y_pred_labels, average='macro')


print("Confusion Matrix:")
print(cm)
print("Recall Score\t", recall)
print("Precision Score\t", precision)
print("Accuracy Score\t", accuracy)
print("F1 Score\t", f1)
print("")
```

```
1424/1424 [==============================] - 21s 15ms/step
Confusion Matrix:
[[1969    0    1 ...    0    0    0]
 [   0 5280   40 ...    0    0    0]
 [   1    1 3208 ...    0    0    0]
 ...
 [   0    0    0 ...    0    0    0]
 [   0    0    0 ...    0    0    0]
 [   0    0    0 ...    0    0    0]]
Recall Score     0.611725171418997
Precision Score  0.6314089018888318
Accuracy Score   0.9790500032940246
F1 Score         0.6085392525655263
```

Figure 5 Calculation of the performance metrics and confusion matrix

Table 1 The performance metrics of the model and their corresponding values.

Performance Metrics	Value
Recall	0.61
Precision	0.63
Accuracy	0.98
F1 score	0.61

3. Results and Discussion

The evaluation of the proposed methodology, its performance, and how it compares to other related works in detecting flood attacks show that the recall obtained is 0.61, the precision obtained was 0.63, the accuracy obtained is 0.98, and the F1 Score is 0.61. The proposed model was tested on the CICIoT2023, and the results of the experiments were analyzed. The Gated Recurrent Unit (GRU) algorithm was implemented along with Python, Keras, and the Sklearn libraries.

3.1. Model's Performance on the CICIoT2023

The CICIoT2023 used for the experiments was the most ideal. This dataset is new and an improvement on its previous versions in terms of size and generalizability. Neto et al.[5] employed an extensive topology of real-world IoT devices to obtain the dataset hence making it very realistic and real-time. It is worth noting that using such a dataset for training the model improves its robustness. Now, with this improved level of robustness and, consequently, reliability, the model can be used in real-world scenarios to add to the security of IoT devices. On training and evaluating the proposed GRU-based model on this improved and realistic dataset for flood attack detection, it was observed that the model performed very well in terms of its accuracy. This method could be the first of several that employ deep learning for detecting flood

attacks on such a real-time dataset. Based on the findings from Table 1, the model suffered a little in its precision, recall, and F1-score. This may be because of the many classes it had to identify and correctly place. However, it had a nearly perfect score in accuracy. This means that the model correctly predicted most of the instances out of the total in the dataset. In simpler terms, the model made more correct predictions and thus enhanced the detection of flood attacks in the IoT environment.

3.1.1. Model Validation Metrics

The performance of the model in detecting the various flood attacks was evaluated using accuracy and model loss. The accuracy of the model measured how effective the model's prediction was as compared to the actual data. The loss function of the model was used to measure its optimality. The loss function also shows the level of errors in the training or validation of the model. Accordingly, a greater loss function denotes a model iteration that underwent poor model optimization, whereas a lower one denotes better model optimization. Figures 6 and 7 show the model's training and validation accuracy and loss function results, respectively.

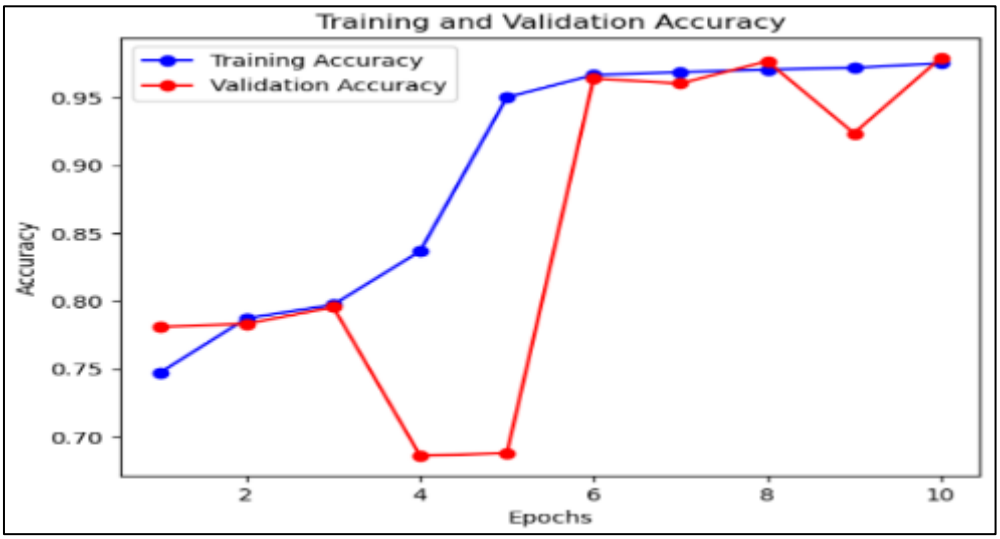


Figure 6 The training and validation accuracy of the model

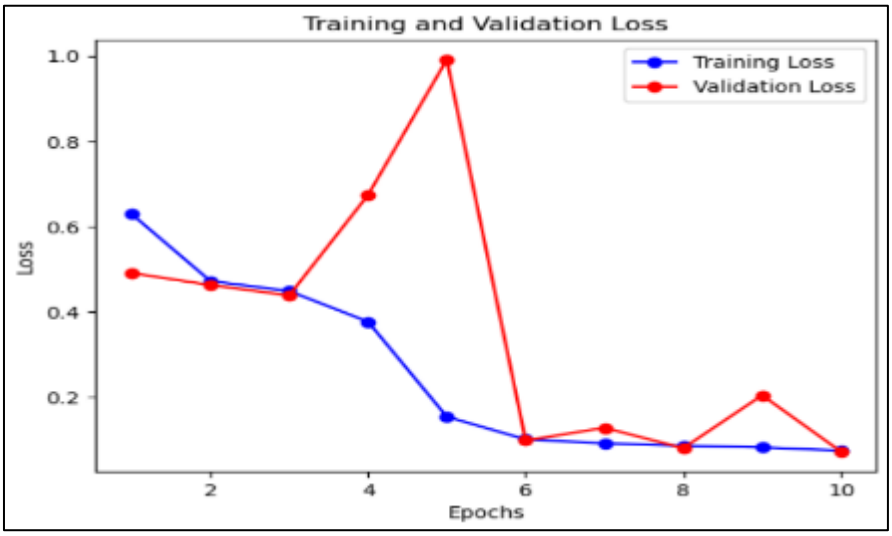


Figure 7 The training and validation loss of the model

In Figure 6, it is seen that the training accuracy steadily rises from the first epoch to the second, slightly falls at the third, and begins a steady rise from there to the fourth epoch. From here it sharply rises to the seventh epoch and then maintains a steady rise through to the tenth epoch. The validation accuracy of the model gently rises from the first epoch to the third. It then sharply falls from this epoch to the fourth while maintaining a steady level until the fifth epoch. The

validation accuracy then very sharply rises at the sixth epoch. From here, there is a slight fall and rise in accuracy between the sixth and the eighth epochs. Then, there is a fall from the eighth epoch to the ninth and a rise to the tenth epoch.

These rises and falls in training and validation accuracy only illustrate the efficiency of the model in detecting flood attack classes during its training and validation over the stipulated number of epochs. For the training accuracy of the model over the epochs, it is seen that there is a consistent rise. This shows that the model kept improving in detecting attacks on data it had already been trained on. However, as it was tested on new data during the validation accuracy, the model initially suffered to efficiently classify the data. Now, after the fifth epoch, the model is seen to greatly pick up in accuracy and maintain a steady rise in accuracy, detecting flood attacks on the new data. In summary, the model performed well and significantly improved in detecting flood attacks on both new data and data it had already been trained on. Inferring from the training and validation accuracy results in Figure 6, the model meets its aim of improving the detection of flood attacks in the IoT environment.

From Figure 7, there is a consistent decrease in training loss from the first to the fourth epoch. Between the fourth and fifth epochs, a sharp fall in loss is recorded. Afterward, there is a steady decrease in loss up until the last epoch. For the validation loss, there is a sharp fall in loss between the first and the third epochs and a very sharp rise to the fifth epoch. From the fifth epoch, there is a steep fall in loss up to the sixth epoch. From there, there is a consistent rise and fall in loss with a short fall at the last epoch.

The training and validation losses are metrics to check for the performance of the model in training. These metrics help to check and prevent overfitting. Overfitting is the state where a model learns and performs very well on the training data instead of learning key patterns in order to perform just as well on new data. This is where validation loss comes in handy. The validation loss helps to evaluate how well the model is performing on never-seen data. Both metrics are key components for optimizing the model in that they help tweak certain model parameters to improve losses. A decrease in training loss and validation loss denotes an improvement in the model's learning on the data it has been trained on and on new data, respectively. As seen in Figure 7, the training loss, from the first to the last iteration or epoch, decreases consistently. This only emphasizes the efficiency of the model. However, the same cannot be said for the validation loss in this same figure. In this figure, when the model was being tested on new data, the validation loss began to alarmingly increase, depicting a decline in its accuracy. Yet, at the fifth epoch, there is a steep decline in validation loss. This shows that the model effectively learned the key patterns necessary for correctly predicting data instances as an attack or not. From there, the validation loss remained at a relative minimum. Thus, illustrating the improvement of the model in rightly placing new data instances of flood attacks in the IoT environment.

Figures 8 and 9, are the ROC and a graphical representation of the performance metrics of the model depicting its overall performance.

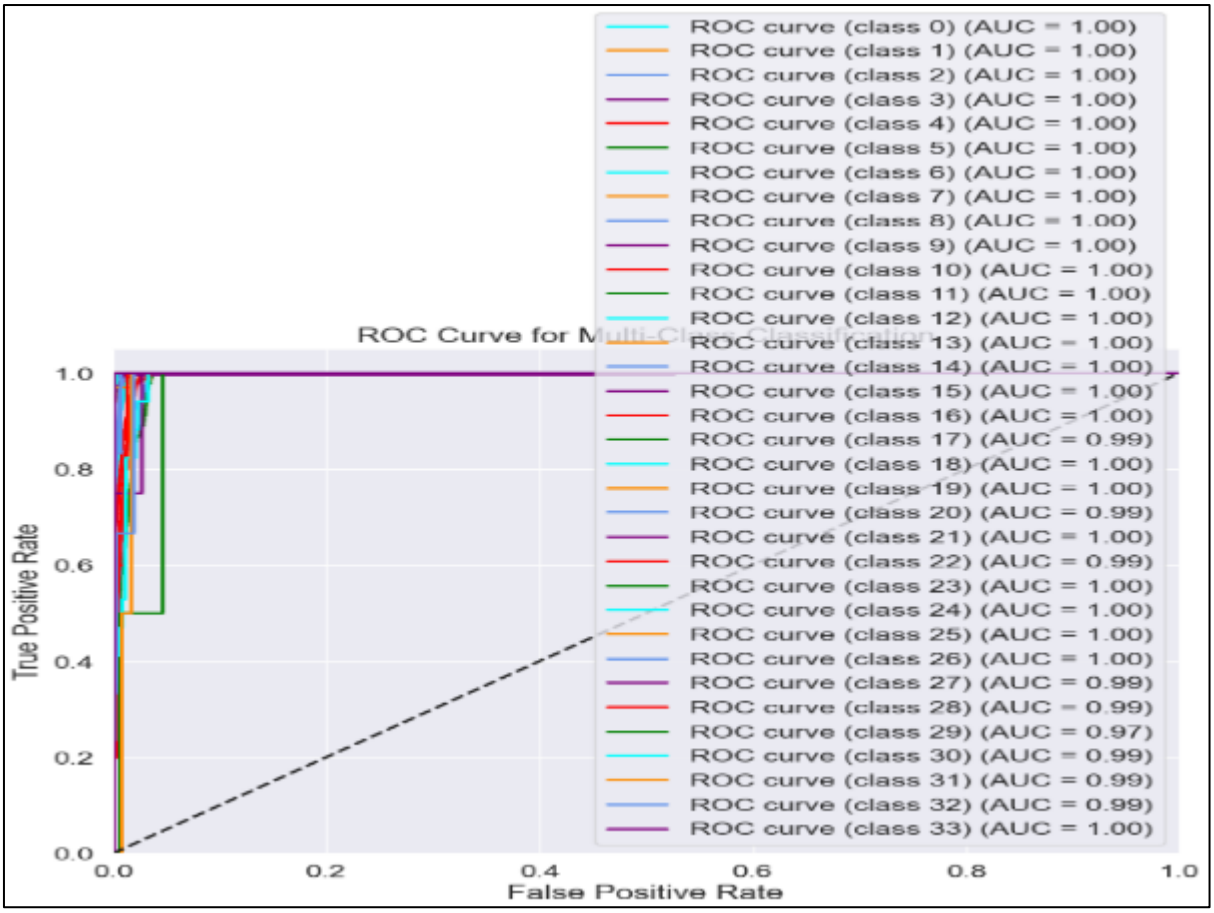


Figure 8 ROC curve for multi-class classification

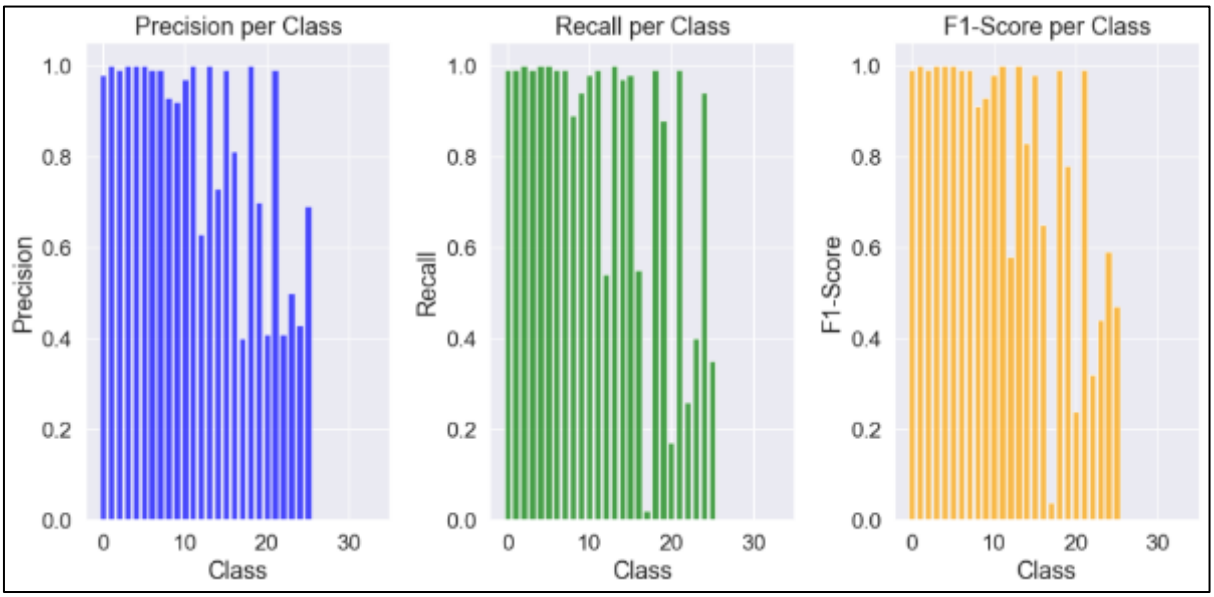


Figure 9 Illustration of the model's performance metrics

As seen in Figure 8, the ROC curve, the AUC score for each class was either 1 or very close to 1. An AUC score equal to 1 is interpreted as perfect performance. Now, from Figure 8, the model is seen to perfectly predict 24 attack classes. This depicts an exceptional performance by the model in detecting the types of flood attacks. The model's AUC score for the other attack classes was at least 0.97, indicating that although it did not perform exceptionally in detecting these classes, it came very close. This level of performance is also reiterated in Figure 9, where the level for F1-score, recall, and

precision in each class of attack shows a good model performance. For each of the classes in relation to precision, recall, and F1-score, the model did very well in detecting most of the classes of flood attacks, as shown by the height of the bars. However, there are certain classes of flood attacks that the model scored poorly across precision, F1-socre, and recall. These are indicated by low bars in Figure 9. Overall, inferring from the accuracy, precision, recall, and F1-score, the model performed well in detecting most of the classes of flood attacks. Thus, enhancing the detection of flood attacks in IoT and adding to the security of IoT devices against such attacks.

3.1.2. Confusion Matrix

The confusion matrix table shows the true positive, true negative, false positive, and false negative data points. The result of the confusion matrix is shown in Figure 10.

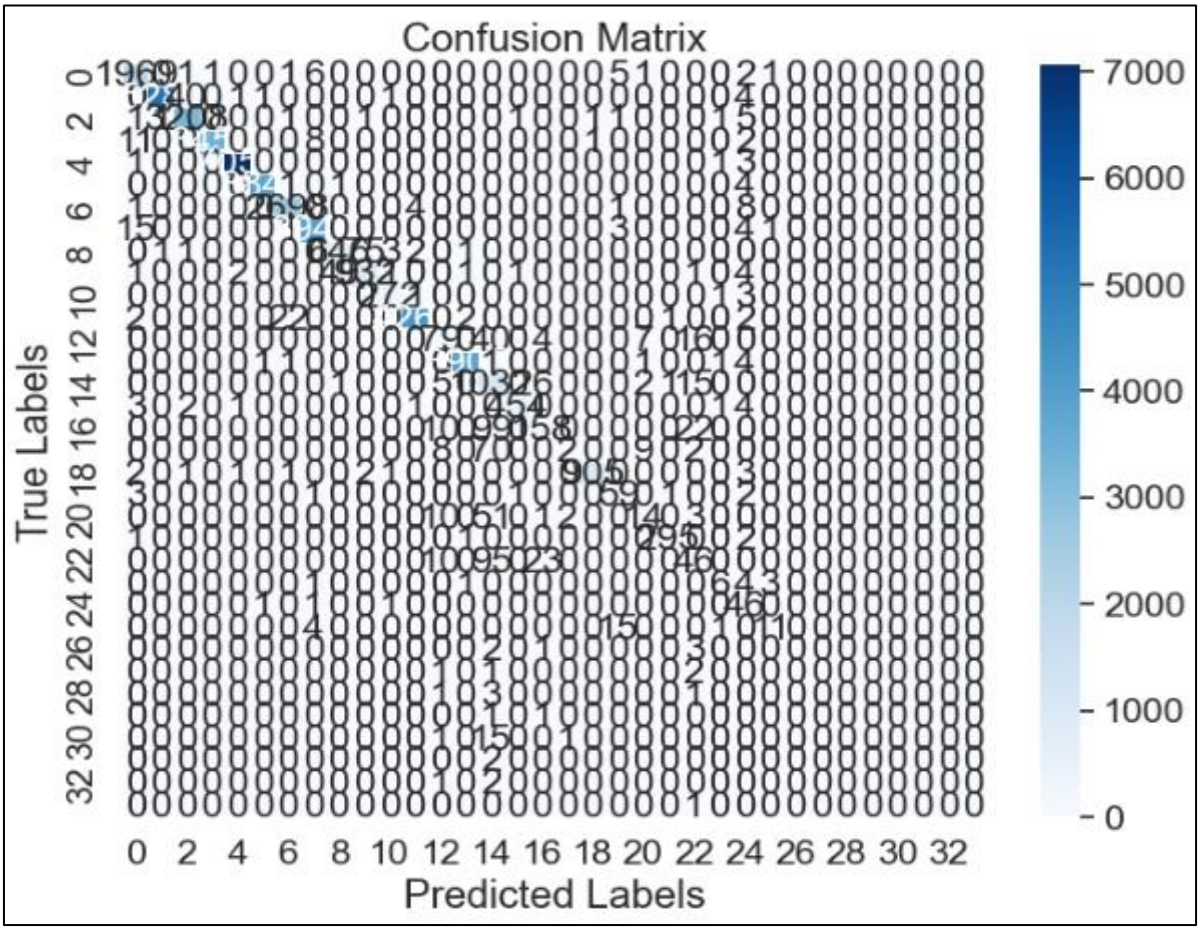


Figure 10. The confusion matrix

It is very difficult to make deductions from this confusion matrix. It looks so because of the multi-class classification nature of this study. The model went through 33 classes of attacks and crammed the results of their confusion matrices.

3.1.3. Comparison of Results with Related Systems

Here, we present a comparison of this work with others based on the available metrics, especially on accuracy. In Table 2, the proposed model is compared to previous works. Contrary to the work of Evmorfos et al.[13]. The proposed GRU model outscored theirs in accuracy. Again, when compared to the revealed metric in the work of Doshi et al. [14], their best AUC score was 0.98, while the proposed system score was 1.00, depicting a perfect performance by the model in detecting most classes of flood attacks. In summary, the model has been shown to improve, in comparison to previous works, the detection of flood attacks in the IoT environment. The analysis and findings of the developed deep learning approach for detecting flood attacks in IoT were discussed. Data cleaning and pre-processing methods were applied to the CIC IoT Dataset 2023. Subsequently, the model underwent training, testing, and validation with accuracy, precision, recall, and F1-score as performance metrics. According to the experiment, the GRU-base model had a near-perfect accuracy score of 0.98. Backed also by the precision, recall, and F1-score values, the model performed well, showing

that the model can be useful in the detection of flood attacks in real-world scenarios. In all, it can be said that the proposed deep learning model enhances the detection of flood attacks in the IoT environment.

Table 2 Comparison of Results with Related Works

Source	Method Used	Dataset Used	Accuracy	AUC
Evmorfos et al.[13]	RNN	Simulated network data	0.81	-
Doshi et al. [14]	Online Discrepancy Test (ODIT)	N-BaIoT	-	0.98
Proposed system	GRU	CICIoT2023	0.98	1.00

4. Conclusion

This study addressed the problem of flood attack detection in IoT environments by developing a deep learning model based on the Gated Recurrent Unit. Trained on the CICIoT2023, which closely resembles real-world IoT traffic, the proposed model demonstrated a high level of accuracy (0.98) and strong potential for real-time application. While the model performed well in terms of overall accuracy, its performance on precision, recall, and F1 score was comparatively lower. This suggests challenges in the classification of individual attack types, likely due to class imbalance and the complexity of multiclass prediction. Reformulating the detection task as a binary classification problem may improve these outcomes. The research answered the primary question concerning how to design a deep learning model that maximizes detection performance while maintaining a low rate of false alarms. It also contributes to the literature by integrating updated datasets with a sequential architecture that captures the temporal nature of attack behavior. The study focuses exclusively on improving the detection of flood attacks and does not consider other security threats that affect IoT environments. The research also does not account for the hardware limitations of IoT devices, which may influence their ability to handle large volumes of network traffic during a flood attack and affect the overall effectiveness of the detection system. In the data processing phase of this study, all attack classes were categorized as malicious, while non-attack classes were categorized as benign. Future research should explore architectural improvements and ensemble learning strategies to enhance detection granularity and generalizability across diverse IoT threat landscapes.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Sadhu, P. K., Yanambaka, V. P., & Abdelgawad, A. (2022). Internet of Things: Security and solutions survey. Sensors, 22(19), 7433. https://doi.org/10.3390/s22197433

[2] Altulaihan, E., Almaiah, M. A., & Aljughaiman, A. (2022). Cybersecurity threats, countermeasures, and mitigation techniques on the IoT: Future research directions. Electronics, 11(20), 3330. https://doi.org/10.3390/electronics11203330

[3] Statista. (n.d.). DDoS attacks number global Q1 2023–Q4 2024. Retrieved from https://www.statista.com/statistics/1557643/ddos-attacks-global-number/

[4] Mishra, N., & Pandya, S. (2021). Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. IEEE Access, 9, 59353–59377. https://doi.org/10.1109/ACCESS.2021.3073408

[5] Neto, E., Dadkhah, S., Zohourian, A., Lu, R., & Ghorbani, A. A. (n.d.). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Retrieved from https://www.unb.ca/cic/datasets/iotdataset-2023.html

[6] Pei, J., Chen, Y., & Ji, W. (2019). A DDoS attack detection method based on machine learning. Journal of Physics: Conference Series, 1237(3), 032040. https://doi.org/10.1088/1742-6596/1237/3/032040

[7] Mliki, H., Kaceam, A., & Chaari, L. (2021). A comprehensive survey on intrusion detection based machine learning for IoT networks. ICST Transactions on Security and Safety, 8(29), 171246. https://doi.org/10.4108/eai.6-10-2021.171246

[8] Ali, A., Mateen, A., Hanan, A., & Amin, F. (2022). Advanced security framework for Internet of Things (IoT). Technologies, 10(3), 60. https://doi.org/10.3390/technologies10030060

[9] Alfatemi, M., et al. (2024). Federated learning approach for DDoS detection in SDN environments. Journal of Cloud Computing.

[10] Stiawan, D., Wahyudi, D., Heryanto, A., Samsuryadi, S., Idris, M. Y., Muchtar, F., Abdullah Alzahrani, M., & Budiarto, R. (2019). TCP FIN flood attack pattern recognition on Internet of Things with rule-based signature analysis. International Journal of Online and Biomedical Engineering, 15(07), 124. https://doi.org/10.3991/ijoe.v15i07.9848

[11] Stiawan, D., Suryani, M. E., Susanto, Idris, M. Y., Aldalaien, M. N., Alsharif, N., & Budiarto, R. (2021). Ping flood attack pattern recognition using a K-means algorithm in an Internet of Things (IoT) network. IEEE Access, 9, 116475–116484. https://doi.org/10.1109/ACCESS.2021.3105517

[12] Lee, S.-H., Shiue, Y.-L., Cheng, C.-H., Li, Y.-H., & Huang, Y.-F. (2022). Detection and prevention of DDoS attacks on the IoT. Applied Sciences, 12(23), 12407. https://doi.org/10.3390/app122312407

[13] Evmorfos, S., Vlachodimitropoulos, G., Bakalos, N., & Gelenbe, E. (2020). Neural network architectures for the detection of SYN flood attacks in IoT systems. In Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments (pp. 1–4). https://doi.org/10.1145/3389189.3398000

[14] Doshi, K., Yilmaz, Y., & Uludag, S. (2020). Timely detection and mitigation of stealthy DDoS attacks via IoT networks. arXiv preprint arXiv:2006.08064. Retrieved from http://arxiv.org/abs/2006.08064

[15] Gönen, S., Barişkan, M. A., Karacayilmaz, G., Alhan, B., Yilmaz, E. N., Artuner, H., & SiNdiRen, E. (2022). A novel approach to prevention of Hello flood attack in IoT using machine learning algorithm. El-Cezeri Fen ve Mühendislik Dergisi. https://doi.org/10.31202/ecjse.1149925

[16] [Alabsi, B., Anbar, M., & Rihan, S. (2023). CNN-CNN: Dual convolutional neural network approach for feature selection and attack detection on Internet of Things networks. Sensors, 23(14), 6507. https://doi.org/10.3390/s23146507

[17] Aktar, S., & Nur, A. Y. (2023). Towards DDoS attack detection using deep learning approach. Computers & Security. https://doi.org/10.1016/j.cose.2023.103251

[18] Kavitha, D., & Ramalakshmi, R. (2024). Machine learning-based DDoS attack detection and mitigation in SDNs for IoT environments. Journal of the Franklin Institute. https://doi.org/10.1016/j.jfranklin.2024.107197

[19] Aktar, S., et al. (2024). Dynamic multi-scale topological representation for enhancing network intrusion detection. Computers & Security.

[20] Modi, A., et al. (2024). Hybrid approach for efficient feature selection in anomaly intrusion detection for IoT networks. Journal of Supercomputing.

[21] Lima Filho, F. S. de, Silveira, F. A. F., de Medeiros Brito Junior, A., Vargas-Solar, G., & Silveira, L. F. (2019). Smart detection: An online approach for DoS/DDoS attack detection using machine learning. Security and Communication Networks, 1–15. https://doi.org/10.1155/2019/1574749

[22] Sarraf, S. (2020). Analysis and detection of DDoS attacks using machine learning techniques. 66(1).

[23] Jia, Y., Zhong, F., Alrawais, A., Gong, B., & Cheng, X. (2020). FlowGuard: An intelligent edge defense mechanism against IoT DDoS attacks. IEEE Internet of Things Journal, 7(10), 9552–9562. https://doi.org/10.1109/JIOT.2020.2993782

[24] Nanthiya, D., Keerthika, P., Gopal, S. B., Kayalvizhi, S. B., Raja, T., & Priya, R. S. (2021). SVM based DDoS attack detection in IoT using IoT-23 Botnet dataset. In 2021 Innovations in Power and Advanced Computing Technologies (i-PACT) (pp. 1–7). https://doi.org/10.1109/i-PACT52855.2021.9696569

[25] Adeshina, Q. A. (n.d.). Machine learning based approach for detecting distributed denial of service attack.

[26] Krishna, R. R., Priyadarshini, A., Jha, A. V., Appasani, B., Srinivasulu, A., & Bizon, N. (2021). State-of-the-art review on IoT threats and attacks: Taxonomy, challenges, and solutions. Sustainability, 13(16), 9463. https://doi.org/10.3390/su13169463

[27] Ramprasath, J., Ramakrishnan, S., Nadarajan, S., Prasanth, D. M., & Kumar, V. S. (2022). Virtual guard against DDoS attack for IoT network using supervised learning method. In 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 1419–1424). https://doi.org/10.1109/ICAC3N56670.2022.10074472

[28] Sarhan, M., Layeghy, S., Moustafa, N., Gallagher, M., & Portmann, M. (2022). Feature extraction for machine learning-based intrusion detection in IoT networks. Digital Communications and Networks. https://doi.org/10.1016/j.dcan.2022.08.012

[29] Amrish, R., Bavapriyan, K., Gopinaath, V., Jawahar, A., & Vinoth Kumar, C. (2022). DDoS detection using machine learning techniques. Journal of ISMAC, 4(1), 24–32. https://doi.org/10.36548/jismac.2022.1.003

[30] Najafimehr, M., Zarifzadeh, S., & Mostafavi, S. A. (2022). A hybrid machine learning approach for detecting unprecedented DDoS attacks. The Journal of Supercomputing, 78(6), 8106–8136. https://doi.org/10.1007/s11227-021-04253-x

[31] Zhong, M., Zhou, Y., & Chen, G. (2021). Sequential model based intrusion detection system for IoT servers using deep learning methods. Sensors, 21(4), 1113. https://doi.org/10.3390/s21041113

[32] Sagu, A., Gill, N. S., & Gulia, P. (2022). Hybrid deep neural network model for detection of security attacks in IoT enabled environment. International Journal of Advanced Computer Science and Applications, 13(1). https://doi.org/10.14569/IJACSA.2022.0130115

[33] Ullah, I., Ullah, A., & Sajjad, M. (2021). Towards a hybrid deep learning model for anomalous activities detection in Internet of Things networks. IoT, 2(3), 428–448. https://doi.org/10.3390/iot2030022

[34] De Souza, C. A., Westphall, C. B., & Machado, R. B. (2022). Two-step ensemble approach for intrusion detection and identification in IoT and fog computing environments. Computers & Electrical Engineering, 98, 107694. https://doi.org/10.1016/j.compeleceng.2022.107694

[35] Yousuf, O., & Mir, R. N. (2022). DDoS attack detection in Internet of Things using recurrent neural network. Computers & Electrical Engineering, 101, 108034. https://doi.org/10.1016/j.compeleceng.2022.108034

[36] Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., & Bahaj, S. A. (2022). Deep learning for intrusion detection and security of Internet of Things (IoT): Current analysis, challenges, and possible solutions. Security and Communication Networks, 1–13. https://doi.org/10.1155/2022/4016073

[37] Ali, M. H., Jaber, M. M., Abd, S. K., Rehman, A., Awan, M. J., Damaševičius, R., & Bahaj, S. A. (2022). Threat analysis and distributed denial of service (DDoS) attack recognition in the Internet of Things (IoT). Electronics, 11(3), 494. https://doi.org/10.3390/electronics11030494

[38] Sharma, B., Sharma, L., Lal, C., & Roy, S. (2023). Anomaly-based network intrusion detection for IoT attacks using deep learning techniques. Computers & Electrical Engineering, 107, 108626. https://doi.org/10.1016/j.compeleceng.2023.108626

[39] Ahmed, G. (2021, February 19). Evaluating deep learning models: The confusion matrix, accuracy, precision, and recall. Retrieved from https://www.kdnuggets.com/2021/02/evaluating-deep-learning-models-confusion-matrix-accuracy-precision-recall.html

[40] Amit, S., Nayak, R., Mishra, D., & Dehuri, S. (2017). A review of clustering techniques and developments. Neurocomputing, 267, 664–681. https://doi.org/10.1016/j.neucom.2017.06.053

[41] Ozcam, B., Kilinc, H. H., & Zaim, A. H. (2021). Detecting TCP flood DDoS attack by anomaly detection based on machine learning algorithms. In 2021 6th International Conference on Computer Science and Engineering (UBMK) (pp. 512–516). https://doi.org/10.1109/UBMK52708.2021.9558989

[42] Statista. (n.d.). Annual number of Internet of Things (IoT) malware attacks worldwide from 2018 to 2023. Retrieved from https://www.statista.com/statistics/1377569/worldwide-annual-internet-of-things-attacks/

[43] Seifousadati, A., Ghasemshirazi, S., & Fathian, M. (n.d.). A machine learning approach for DDoS detection on IoT devices.

[44] Procopiou, A., Komninos, N., & Douligeris, C. (2019). ForChaos: Real time application DDoS detection using forecasting and chaos theory in smart home IoT network. Wireless Communications and Mobile Computing, 1–14. https://doi.org/10.1155/2019/8469410

[45] Brownlee, J. (2022, August 19). How to calculate precision, recall, F1, and more for deep learning models. Retrieved from https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/

[46] Pei, W., et al. (2019). A survey on network intrusion detection techniques for DDoS attacks. Journal of Computer Networks and Communications.