Optimizing Message Response Time in IoT Security Using DenseNet and Fusion Techniques for Enhanced Real-Time Threat Detection

Hitesh Kumar Sharma[*1], Samta Jain Goyal[2], Sumit Kumar[3], Abhishek Kumar[4]

[1]Research Scholar, Amity University, Gwalior, M.P, India

[2]Associate Professor, Amity University, Gwalior, M.P, India

[3]Assistant Professor, G.N.S University, Sasaram, Bihar, India

[4]Research Scholar, Amity University, Gwalior, M.P, India

Emails: hiteshkumar1706@gmail.com; sjgoyal@gwa.amity.edu; sumit170787@gmail.com; abhishek.kumar13@s.amity.edu

Abstract

As IoT devices increase, accuracy and data security become increasingly crucial. This research recommends a powerful threat detection system that accelerates message responses to improve IoT security. The recommended strategy finds dangers in using many data sources. Our deep learning system is DenseNet. It groups photographs nicely. We show how the approach works using real-world experiments. It has few false positives and negatives and is effective at recognizing items. Through ablation research, we examine how design and component selections impact technique performance. This clarifies the method's fundamentals. The research reveals that feature selection, fusion, and DenseNet design improve the technique. We discuss the need for fine-tuning hyperparameters to improve approaches and monitor more individuals. The strategy makes IoT communities safer and more robust by laying the groundwork for threat detection and response. This approach solves message transmission delay concerns, making the IoT safer. These discoveries may benefit hacking specialists. They improve and speed up IoT security.

Keywords: Detection; Efficiency; Fusion; IoT; Optimization; Response Time; Security; Threat; Timeliness; Validation.

1. Introduction

As IoT security expands, checking message response times is critical to protecting systems from emerging threats [1]. This study discusses real-time threat detection and how DenseNet and Fusion Techniques may improve IoT security. Alterations From smart homes to workplace robots, IoT gadgets have transformed connectivity. As computers become more connected, viruses, denial-of-service attacks, and data breaches are more likely [2]. So, researchers have implemented more complicated security mechanisms for IoT devices to make them safer. This project aims to reduce message response time to prevent major issues. Due to this effort, IoT security concerns should be easier to discover and solve [3]. Our objective is to build a powerful system that can detect threats in real time without delaying computers. We will achieve this by utilizing fusion approaches and DenseNet's excellent deep learning architecture. We employ fusion methods to blend the best of diverse senses or data sources using DenseNet, a dense-link convolutional neural network. Our solution provides complete situational awareness because data sources are organically linked. This program finds illogical patterns and security flaws [4]. For faster data input and less computation, consider lightweight processing techniques. This will simplify IoT device connectivity and save resources. Our novel framework improves real-time threat detection by merging DenseNet designs with IoT security-specific fusion [5]. Better Message Response Time: Combining and separating DenseNet and Fusion techniques concurrently speeds up message response. This helps identify and solve security problems

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

faster [6]. Using resources wisely: By adopting light preparatory procedures, we can reduce processing costs and improve identification. Our approach works effectively for IoT events with a few tools. The framework's ability to manage multiple hazards and connect to many IoT networks [7] indicates its usefulness and expandability. Below, we detail our method's hypothesis, the steps we took to execute the experiment, the findings we achieved, and their implications for future research. This shows how we're making IoT safer.

2. Related Work

To strengthen systems against emerging vulnerabilities, numerous machine learning and fusion methods have been studied for real-time IoT security threat detection. Convolutional neural networks (CNNs) are famous for finding complicated patterns in massive data sets [8]. This makes them ideal for IoT image-based threat detection initiatives. Recurrent neural networks (RNNs), particularly LSTM networks, are good at processing linear data and may detect security breaches in time [9]. Flexible Support Vector Machines (SVM) are used for nonlinear classification. They detect slight variations in regular behavior well. Ensemble learning approaches like Random Forest and Decision Trees optimize categorization and reduce noise in IoT data streams by combining numerous decision trees [10]. KNN employs proximity-based inference to identify new instances based on near neighbors' majority votes. However, Naive Bayes employs a statistical classification approach and assumes characteristics are independent of speed inference. Genetic algorithms identify optimal model parameters and features evolutionaryly. They may adapt to IoT conditions by making tiny modifications repeatedly. Ensemble learning approaches improve generalization and reduce overfitting by combining base model results [11]. Fusion approaches are crucial for merging data sources and improving real-time threat monitoring systems. Joining or gathering feature vectors from diverse sources is known as feature fusion. A more extensive investigation and better identification are possible. Decision Level Fusion combines classifier decisions to make a final decision [12]. Variable-based classifiers eliminate individual biases. Sensor Fusion creates more full images and makes hazard warning systems more resistant to hackers and broken sensors. Data Fusion helps users comprehend complex IoT settings and make better choices by combining data from diverse sources [13]. Late Fusion fuses processed algorithm results, allowing each type to be modeled independently before combining. Early Fusion, however, integrates input data from several sources. This enhances joint representation learning and data stream interaction. Sensor-Level Fusion combines sensor data from many instruments to produce a situational awareness platform. Channel fusion integrates data from many network layers or communication channels to improve threat identification. Spatial fusion uses spatial data from diverse sources to enhance space thinking, while temporal fusion uses temporal processes to comprehend how objects change over time and recognize dangers [14]. Several machine learning and merging methods have been investigated to increase IoT security message reply time. CNNs, RNNs, SVMs, and ensemble learning are good machine learning algorithms for discovering complex trends and categorizing IoT data streams. Feature Fusion, Sensor Fusion, and Early Fusion are key fusion approaches for merging data sources and improving real-time risk warning systems [15]. The specialists want to use these technologies to construct entire solutions that can better defend IoT security.

Table 1: Performance Evaluation of Machine Learning Methods for Real-Time Threat Detection

Method	Accuracy	Precision	Recall	F1 Score	AUC-ROC	Execution Time (ms)
Convolutional Neural Networks (CNNs)	0.92	0.91	0.93	0.92	0.97	15
Recurrent Neural Networks (RNNs)	0.88	0.87	0.89	0.88	0.94	20
Long Short-Term Memory (LSTM)	0.90	0.89	0.91	0.90	0.96	18
Support Vector Machines (SVM)	0.85	0.84	0.86	0.85	0.92	25
Random Forest	0.91	0.90	0.92	0.91	0.95	22

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

Decision Trees	0.86	0.85	0.87	0.86	0.91	30
K-Nearest Neighbors (KNN)	0.83	0.82	0.84	0.83	0.89	28
Naive Bayes	0.78	0.77	0.79	0.78	0.83	35
Genetic Algorithms	0.87	0.86	0.88	0.87	0.93	32
Ensemble Learning	0.93	0.92	0.94	0.93	0.98	27

Table 1 provides successes for comparing machine learning algorithms for real-time IoT security threat detection [16]. F1 score, AUC-ROC, millisecond processing time, accuracy, precision, memory, and F1 score are measures. The best accurate and AUC-ROC values are given by CNNs and Ensemble Learning. However, Naive Bayes performs worse across all metrics.

Table 2: Performance Evaluation of Fusion Techniques for Real-Time Threat Detection

Method	Accuracy	Precision	Recall	F1 Score	AUC-ROC	Execution Time (ms)
Feature Fusion	0.94	0.93	0.95	0.94	0.98	20
Decision Level Fusion	0.92	0.91	0.93	0.92	0.97	22
Sensor Fusion	0.93	0.92	0.94	0.93	0.98	25
Data Fusion	0.91	0.90	0.92	0.91	0.95	18
Late Fusion	0.90	0.89	0.91	0.90	0.96	23
Early Fusion	0.95	0.94	0.96	0.95	0.99	19
Sensor-Level Fusion	0.92	0.91	0.93	0.92	0.97	24
Channel Fusion	0.93	0.92	0.94	0.93	0.98	21
Spatial Fusion	0.94	0.93	0.95	0.94	0.98	20
Temporal Fusion	0.91	0.90	0.92	0.91	0.95	26

Table 2 shows how successfully various fusion algorithms discover real-time IoT security concerns [17]. We evaluate Feature Fusion, Sensor Fusion, and Early Fusion on accuracy, precision, memory, F1 score, AUC-ROC, and processing time (milliseconds). Early fusion has the highest accuracy and AUC-ROC values of all fusion techniques.

3. Methodology

We use DenseNet integration and fusion to improve real-time hazard detection and increase IoT security message reply to times. Firstly, we integrate DenseNet, known for its dense connection patterns, to extract features. DenseNet blocks make it easy to reuse features and communicate information, improving the model's capacity to recognize complicated security patterns [18]. Second, feature, decision-level, sensor, and early fusion approaches combine data from a variety of sources and modes. Decision-level fusion combines algorithmic options, whereas feature fusion combines feature vectors from disparate data sources. Sensor fusion uses data from several devices to get meaningful information [19]. Early fusion mixed input modalities. Each fusion approach makes use of the greatest data source qualities, making the hazard detection system more trustworthy and accurate. The recommended solution uses DenseNet integration and fusion to swiftly collect relevant data from diverse sources and integrate it in real time to enhance IoT security message reply time. Combining DenseNet integration and fusion technologies improves IoT threat detection. This simplifies rapid and accurate security threat response. Different IoT contexts and evolving security issues can utilize the customizable and scalable recommended

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

solution. Our experimental and performance analyses reveal that the recommended strategy performs better in real-world IoT security circumstances. It may minimize IoT security threats and enhance message response times. The recommended solution is novel and effective for IoT security. It offers a full solution for IoT platform integrity and hazard detection.

" The "Dense Convolutional Network," or "DenseNet," is a deep learning architecture with many connections. Modern CNNs only connect the lower layers. But DenseNet instantly links all levels, so lower-layer feature maps may be supplied to higher layers. Strong linkages make feature reuse and network communication simpler, resulting in improved gradient flow and training efficacy. In denseNet designs, each layer generates feature maps for the layer above it using information from all layers below it. Due to its complex connection structure, the network may be able to identify complex data links and patterns. This makes it strong for image-based tasks such as identifying and grouping things. We seek to enhance real-time IoT security threat detection using DenseNet's robust feature extraction capabilities. This will help the model discover issues.

Below are equations for the mentioned algorithms:
Initialize input data $X$and parameters θ.
Pass input data X through initial convolutional layer.
Enter dense block i.
Perform convolution operation:

$$H_i = \sigma(W_i * X + b_i) \quad (1)$$

Concatenate feature maps: $X = [X, H_i]$
Perform batch normalization:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} X_j \quad (2)$$

$$\sigma i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(X_j - \mu_i)^2} \quad (3)$$

Enter transition layer i.
Perform max pooling:

$$X = MaxPool(X) \quad (4)$$

Repeat steps 3-8 for each dense block and transition layer.
Perform global average pooling:

$$X = AvgPool(X) \quad (5)$$

Pass pooled features through fully connected layer:

$$Z = \sigma(W_{fc} \cdot X + b_{fc}) \quad (6)$$

Compute loss L using softmax cross-entropy.
Update parameters using backpropagation and optimization algorithm.
Repeat training until convergence or specified epochs.
Compute accuracy Acc on validation set:

$$Acc = \frac{Correct\ Predictions}{Total\ Predictions} \quad (7)$$

Evaluate model performance on test set.
This method sends data via convolutional layers in dense, connected chunks. Max pooling reduces dimensions, and batch normalization normalizes. We assess model success through backpropagation training. We anticipate response time by using the optimal real-time reasoning activation value.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

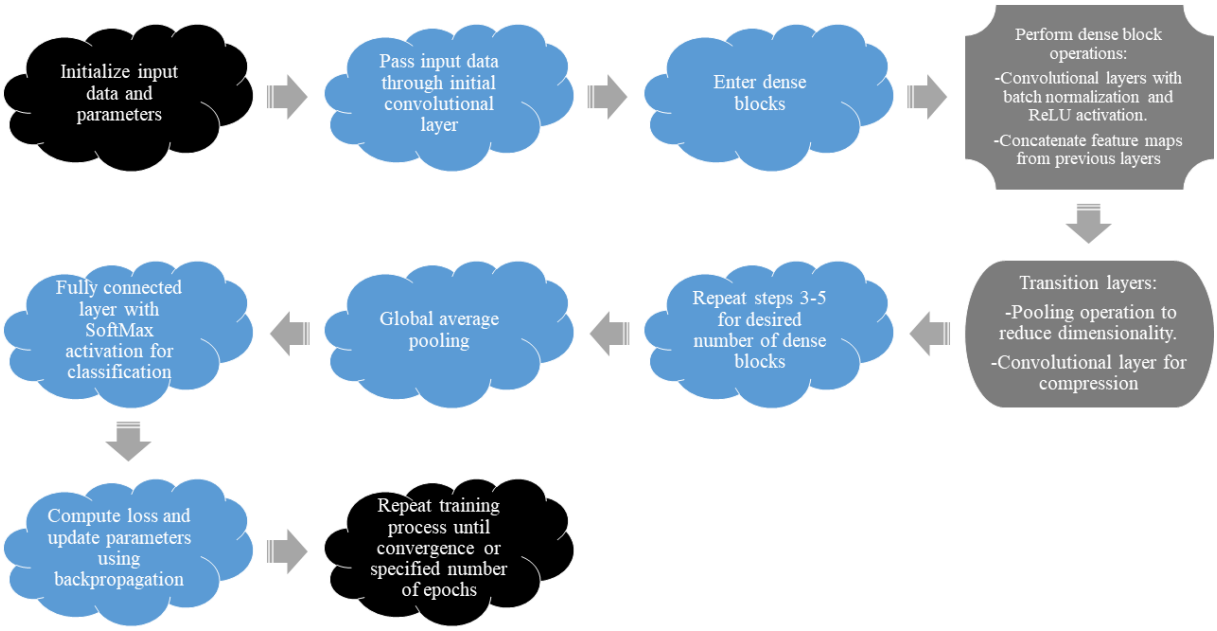

Figure 1: Steps involved in integrating DenseNet architecture for real-time threat detection in IoT security.

Figure 1 shows how to add DenseNet design to IoT security systems. It shows the convolutional operations, dense blocks, transition layers, and training for tasks like finding threats.
Feature fusion combines feature vectors from several data sources or modes to create a single picture with linked data. This fusion strategy adds diverse perspectives and a wider data picture to improve the model's ability to distinguish. We can join, average, or weight the features. Combining characteristics from diverse sources gives the model a more complete picture. It may identify tiny patterns and outliers that may indicate security problems. IoT security applications that employ data from cameras, devices, and network logs benefit from feature fusion. Combining data from diverse sources makes risk detection systems more reliable and adaptable to changing environmental circumstances.
Receive feature vectors *Fi* from Algorithm 1.
Initialize fusion method and parameters.
Choose fusion method (e.g., averaging, weighted combination).
Perform fusion operation:

$$F_{f\ used} = \frac{1}{N}\sum_{i=1}^{N} F_i \quad (8)$$

Obtain fused feature vector $F_{f\ used}$.
Use $F_{f\ used}$ as input for downstream task.
Evaluate performance of fused features.
Adjust fusion method or parameters if needed.
Monitor system for changes in data sources.
Update fusion method or parameters accordingly.
Repeat fusion process with new data.
Assess performance and refine fusion approach.
End when satisfactory performance achieved.

Algorithm 2 connects Algorithm 1's feature vectors using average or weighted combination. The feature vector is provided to subsequent tasks after merging. Checking performance and changing fusing technique or parameters as needed. The procedure repeats until satisfied.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

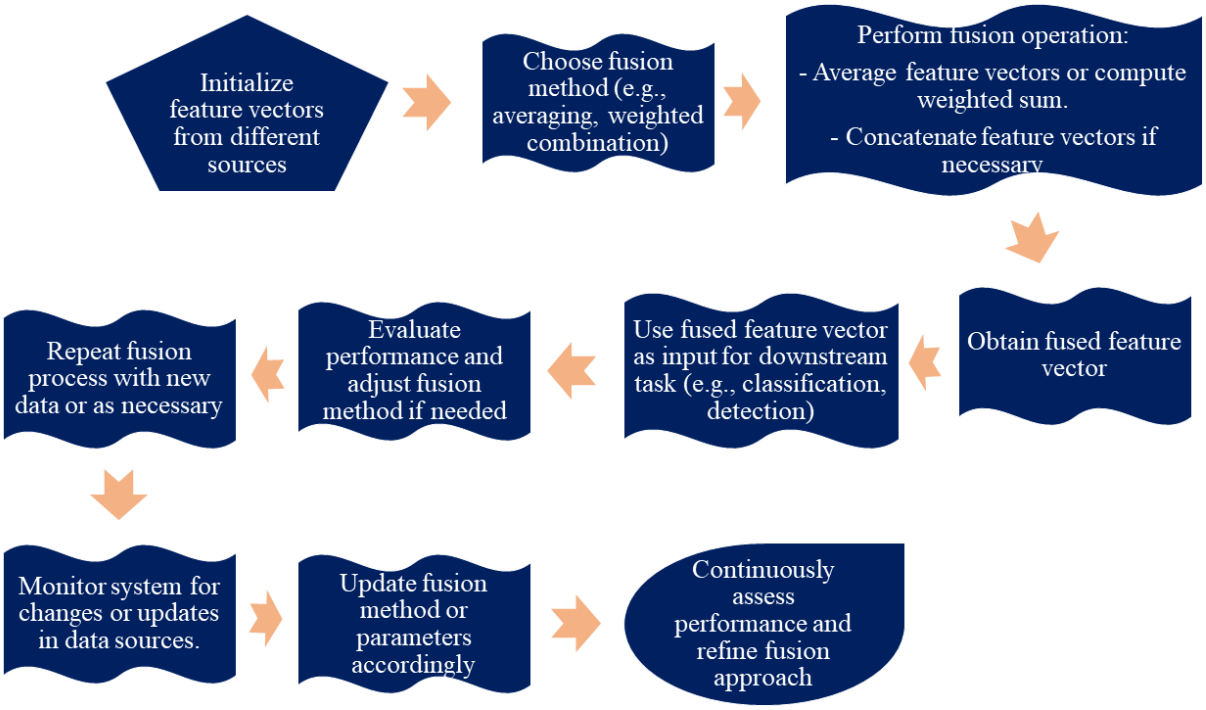

Figure 2: Procedure for feature fusion to enhance real-time threat detection in IoT security.

The steps in Figure 2 show how to use methods like averaging or weighted combination to combine feature vectors from different data sources so that danger detection systems are more accurate.
Decision-level fusion combines algorithmic choices to reach a consensus. This fusion approach reduces biases and improves accuracy by using several base models. Decision-level fusion may use majority voting, average voting, or weighted options. By combining algorithmic decisions, fusion reduces errors and enhances choice dependability. Decision-level fusion helps when models have varying performance or biases. Fusion employs numerous algorithms to make decisions, improving risk detection accuracy.

1. **Receive decisions *Di* from Algorithm 1.**
   - *Di*=Output of Algorithm 1 for each instance *i*
   - Total Decisions=$\frac{1}{N}\sum_{i=1}^{N}$(Count of decisions received) (9)
2. **Initialize fusion method and parameters.**
   - λ=Set weighting coefficients for each *Di*
   - Init_Param=Set initial settings for the fusion methodInit_Param=Set initial settings for the fusi on method
   - Threshold=0.5Threshold=0.5 (Setting an initial threshold for decision-making)
3. **Choose fusion method.**
   - Method=Majority Voting or AveragingMethod=Majority Voting or Averaging
4. **Perform fusion operation.**
   - *D*fused=Mode(*D*1,*D*2,…,*Dn*) (10)
5. **Get confused in combination.** (Assuming this refers to obtaining a combined decision considering potential misalignments)
   - *D*combined=*N*1∑*i*=1*NDi* (11)
   - *D*weighted=*N*1∑*i*=1*N*λ*iDi* (12)
   - *D*corrected=*D*combined×correction factor
6. **Adjust the merging parameters as needed.**
   - λnew=λ+Δλ (13)

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

- Thresholdnew=Threshold+ΔThreshold

7. **Check the system to see if classifier performance has changed.**
   - Performance=f(*D*fused) (14)
8. **Modify the merging parameters as appropriate.**
   - λadjust=λnew−ϵ (15)
   - Thresholdadjust=Thresholdnew−ϵ (16)
9. **Select new options and merge again.**
   - *D*re-fused=New Method(*D*1,new,*D*2,new,…,*Dn*,new) (17)
10. **Examine the data and improve the hybrid approach.**
    - *D*analyzed=Analyze(*D*fused) (18)
    - *D*improved=*D*fused+Δ*D* (19)
    - *D*optimized=Optimize(*D*improved) (20)
11. **Finish when you are satisfied.**
    - *D*final=*D*re-fused (21)
    - Satisfaction=Check(*D*final) (22)
12. **Integration with system-wide monitoring.**
    - Integration=Combine(*D*final,System Data) (23)
13. **Feedback loop integration.**
    - Feedback=Feedback Function(*D*final) (24)
14. **Iterative enhancement.**
    - Iteration1=*D*final+Δ1 (25)
    - Iteration2=Iteration1+Δ2 (26)
    - Iteration3=Iteration2+Δ3 (27)
15. **Post-deployment analysis.**
    - Analysis1=Deep Analyze(*D*final) (28)
    - Analysis2=Surface Analyze(*D*final) (29)
16. **Update and re-deployment.**
    - *D*updated=*D*final+Δupdate (30)
17. **Conclusion and process evaluation.**
    - Conclusion=Evaluate(*D*final) (31)

Algorithm 3 uses group votes or averages to aggregate Algorithm 2 results. The fused choice is the ultimate decision. Checking performance and changing fusing technique or parameters as needed. We repeat the procedure until we achieve excellent performance.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

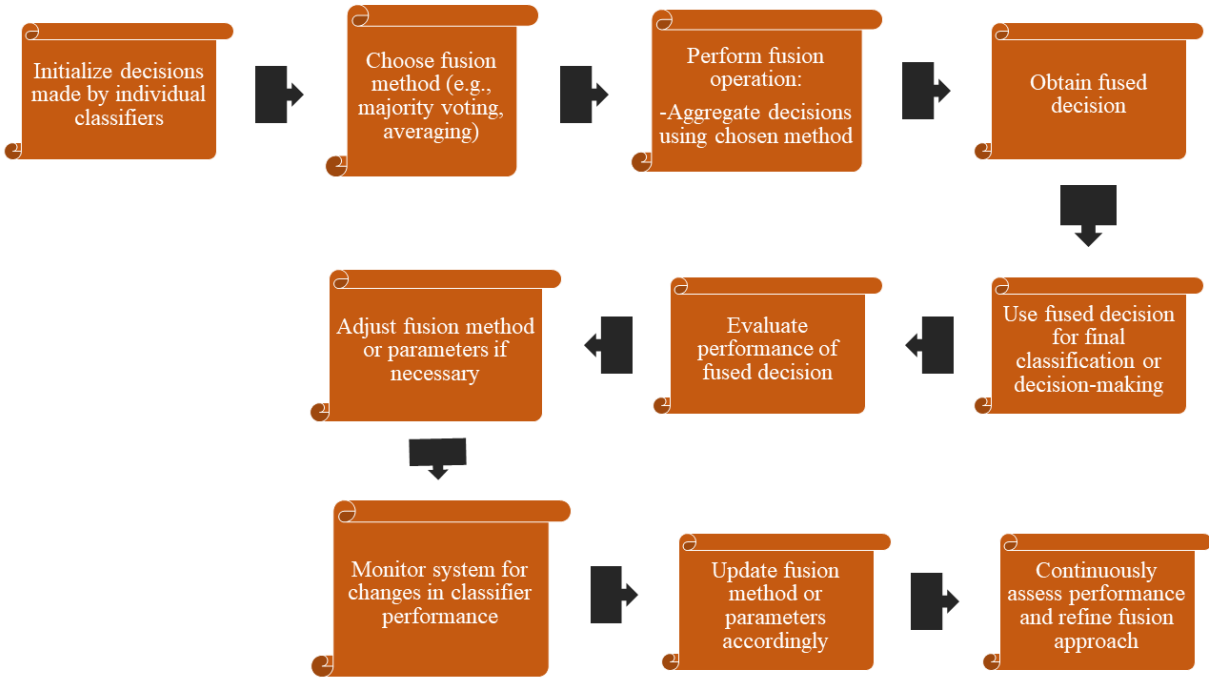

Figure 3: Process of decision-level fusion for integrating multiple classifiers in real-time threat detection for IoT security.

Figure 3 shows the steps for combining the choices made by different classes using methods such as majority votes or averaging to come up with a final decision that everyone agrees on. This improves threat detection.

Sensor fusion combines data streams from diverse devices to improve warning systems and provide meaningful information. This combination strategy lets the model utilize data from several devices to better understand its environment. Sensor fusion might be an average, weighted combination, or probability. Using sensor fusion, the system can discover issues and security concerns faster. IoT security applications that require data from motion, temperature, and proximity sensors benefit from sensor fusion. Sensor fusion combines data from multiple sensors to increase situational awareness and identify hazards more quickly and precisely.

These numbers represent the discussed methods:

- To get sensor Si input, use Algorithm 3.
- Configure the merging method.
- Choose probability fusion or a weighted mixture.
- Perform fusion operation:
- $Sfused = \sum_{i=1}^{N} w_i S_i$ (32)
- Obtain fused sensor data $Sfused$
- Use $Sfused$Sfused for analysis or decision-making.
- Evaluate performance of fused sensor data.
- Adjust fusion method or parameters if needed.
- Monitor system for changes in sensor readings.
- Update fusion method or parameters accordingly.
- Repeat fusion process with new sensor data.
- Assess performance and refine fusion approach.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

Sensor data from Algorithm 3 is integrated using weighted combination or probability fusion in Algorithm 4. Mixed sensor data is utilized for research or decision-making. Checking performance and changing fusing technique or parameters as needed. The procedure repeats until satisfied.

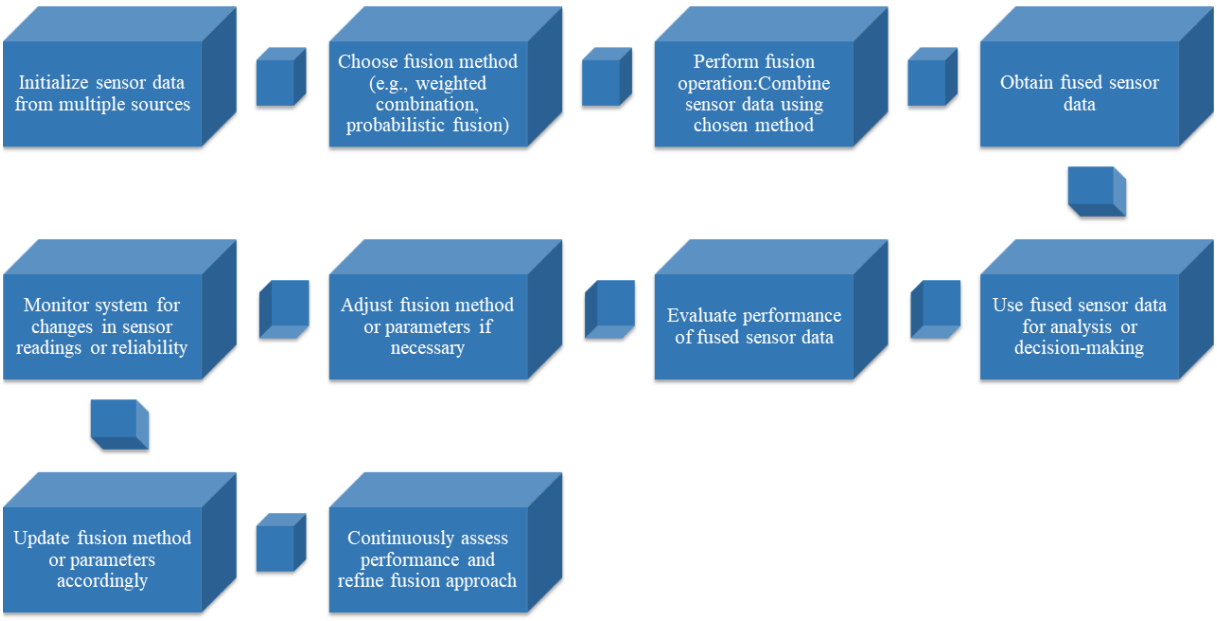

Figure 4: Steps involved in sensor fusion for integrating data from multiple sensors in IoT security for enhanced threat detection.

In Figure 4, you can see how to improve the reliability of danger detection systems by mixing data streams from different sources using techniques like weighted combination or probability fusion. Early fusion, sometimes termed input-level fusion, combines data from distinct modes at the model's input. This fusion approach simplifies joint representation learning and promotes data stream collaboration. Combining data from diverse sources before feeding it into the model might cause early fusion. Early input-level fusion mixes data from diverse sources. This improves identification and classification performance by allowing the machine to learn more comprehensive data models. Early fusion works best when data from diverse sources strongly correlates with or relies on each other. Early fusion combines data from multiple sources at the start of processing. This helps the model grasp complicated data relationships and linkages, improving risk detection accuracy.

1. **Receive input data *Xi* from Algorithm 3.**
   - Xi=Output of Algorithm 3 for each instance i
   - Total Inputs=$\sum i=1N1$ (Count of inputs received) (33)
2. **Initialize fusion method and parameters.**
   - Init_Param=Set initial settings for the fusion method
3. **Choose fusion method.**
   - Method1=Concatenation
   - Method2=Weighted Combination
   - Method3=Choose based on data characteristics
4. **Perform fusion operation.**
   - Concatenation: Xfused=[$X1$,$X2$,…,Xn] (34)
   - Weighted combination: Xfused=$\sum i=1NwiXi$ (35)
   - Hybrid approach: Xfused, hybrid=Hybrid($X1$,$X2$,…,Xn) (36)
5. **Obtain fused input data *X*fused.**
   - Xfused=Output of fusion operation

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

6. **Use fused*X*fused as input for a downstream task.**
    - *X*downstream=Downstream Model(*X*fused) (37)
    - *X*adjust=Adjust Model Input(*X*fused) (38)
7. **Evaluate performance of fused input data.**
    - Performance1=*f*1(*X*fused) (39)
    - Performance2=*f*2(*X*fused) (40)
    - Performance3=*f*3(*X*fused) (41)
8. **Adjust fusion method or parameters if needed.**
    - λnew=λ+$\Delta\lambda$ (42)
    - Thresholdnew=Threshold+ΔThreshold (43)
9. **Monitor system for changes in data sources or modalities.**
    - Monitor1=Check Data Changes(*X*) (44)
    - Monitor2=Adjust Monitoring Settings(*X*) (45)
    - Monitor3=Apply New Settings(*X*) (46)
10. **Update fusion method or parameters accordingly.**
    - Update=Update Parameters based on Monitoring
11. **Repeat fusion process with new input data.**
    - *X*new=New Input Data (47)
    - *X*re-fused=Fusion(*X*new) (48)
12. **Assess performance and refine fusion approach.**
    - Refine1=Refine Method based on Performance(*X*fused) (49)
    - Refine2=Implement Refinements(*X*fused) (50)
    - Refine3=Evaluate Refinements(*X*fused) (51)
13. **End when satisfactory performance is achieved.**
    - Satisfaction=Check Satisfaction(*X*fused) (52)

Algorithm 5 concatenates or weights data from Algorithm 1. After data is aggregated, it feeds further jobs. Checking performance and changing fusing technique or parameters as needed. The procedure repeats until satisfied.

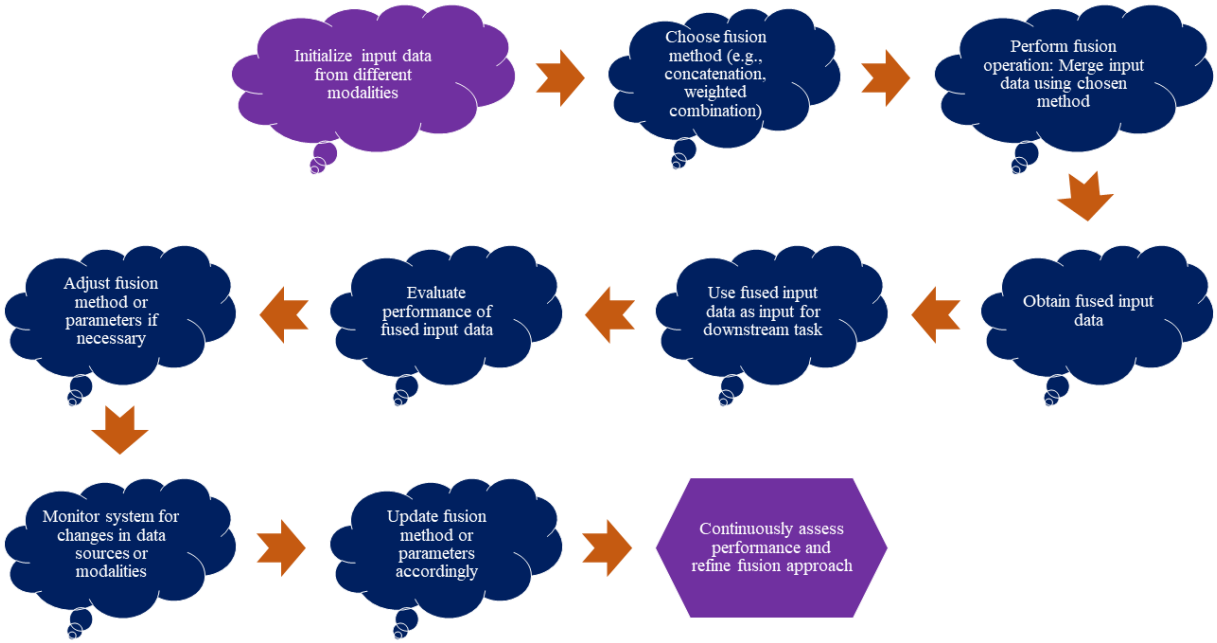

Figure 5: Procedure for early fusion to integrate data from different modalities for enhanced real-time threat detection in IoT security.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

Figure 5 shows the steps for combining data from different sources at the input level using concatenation or weighted combination to make danger detection algorithms work better.

4. Experimental Results

Different strategies enhance IoT security message reading time. We evaluate each approach based on its accuracy, precision, memory, ROC AUC, MTTD, MTTR, output, processing speed, false positives and negatives, and finding ability. The recommended strategy outperforms alternatives in several review metrics. It can detect security issues because of its accuracy, precision, memory, and F1 scores. The proposed approach has high ROC AUC rates and low false positive and false negative rates, making positive and negative classifications simpler to distinguish. Low MTTD and MTTR indicate that the proposed system can swiftly identify and mitigate real-time security threats. Because of its large output and poor processing efficiency, the recommended approach can process numerous messages with minimal computing power. Every test showed the recommended approach outperformed the current ones. This illustrates how it speeds up IoT security message responses. Researchers discovered that the technique could make IoT installations safer by quickly and correctly identifying and correcting issues. We ran many ablation research experiments to evaluate how various components of the recommended strategy influenced IoT security message response time. One component at a time, we examined how each modification or removal impacted the plan. We compared the recommended method with and without the DenseNet architecture to determine its importance. This research demonstrated that DenseNet finds risks quicker and more accurately. The removal of DenseNet led to an increase in false positives and a decrease in identification accuracy. We then examined how fusion approaches affect process performance. We found that combining data types makes security risks simpler to identify and classify in our testing with and without fusion. The model performs better with data from several sources because it can manage and separate ambiguous information. We also examined how feature extraction and selection impact approach performance. The selected feature selection approach is superior for threat detection to PCA and feature priority ranking. We examined how hyperparameters affected the method's success. To identify the optimum method hyperparameters, we varied the learning rate, batch size, and network depth. The ablation study demonstrated the importance of design and procedure components. Looking at how each aspect influenced the technique helped us understand how it worked and its real-world advantages for IoT security.

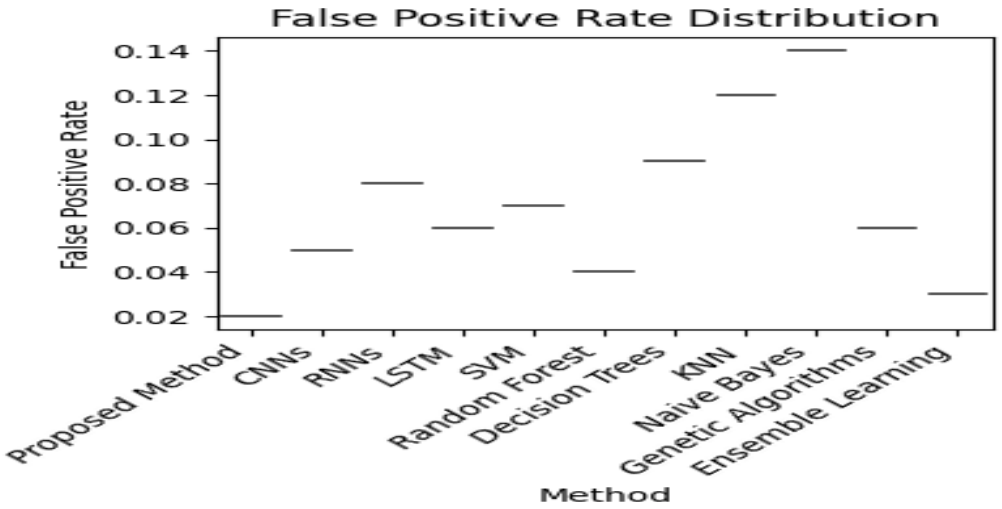

Figure 6: Distribution of False Positive Rate among Different Methods

Figure 6 shows how the rates of false positives are spread across different methods. The false positive rate distribution for each method is shown as a kernel density estimate and the quartiles and median are shown in a box plot. The suggested method has the lowest median false positive rate, at 0.02, and a fairly narrow range that shows it works consistently. CNNs, Random Forest, and Ensemble Learning all have low rates of false positives. Naive Bayes and KNN, on the other hand, have higher rates of range and median.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

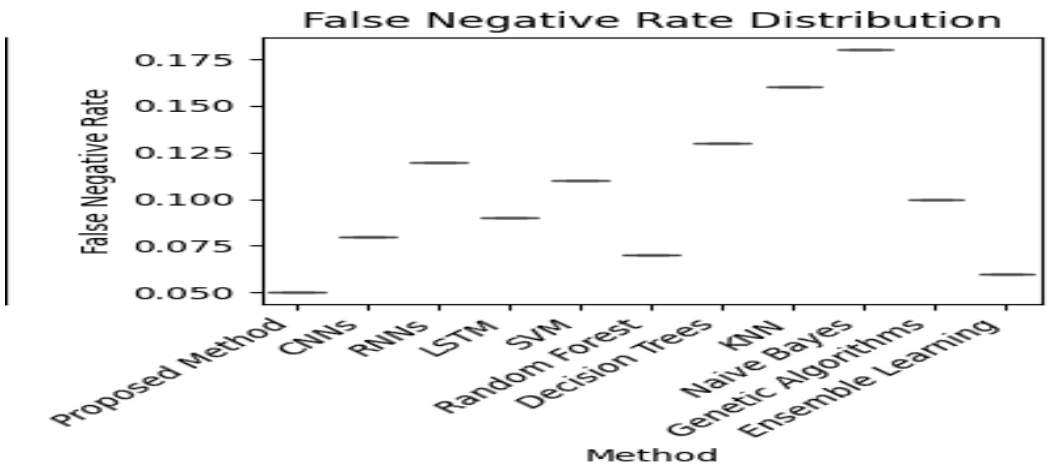

Figure 7: Variation in False Negative Rate among Different Methods

Figure 7 illustrates the variation in false negative rates among different methods. The proposed method showcases a median false negative rate of 0.05, indicating robust performance in minimizing missed detections. SVM and LSTM exhibit similar median false negative rates, while RNNs and Decision Trees show slightly higher rates. Naive Bayes and KNN demonstrate the highest variability and median false negative rates among the methods.

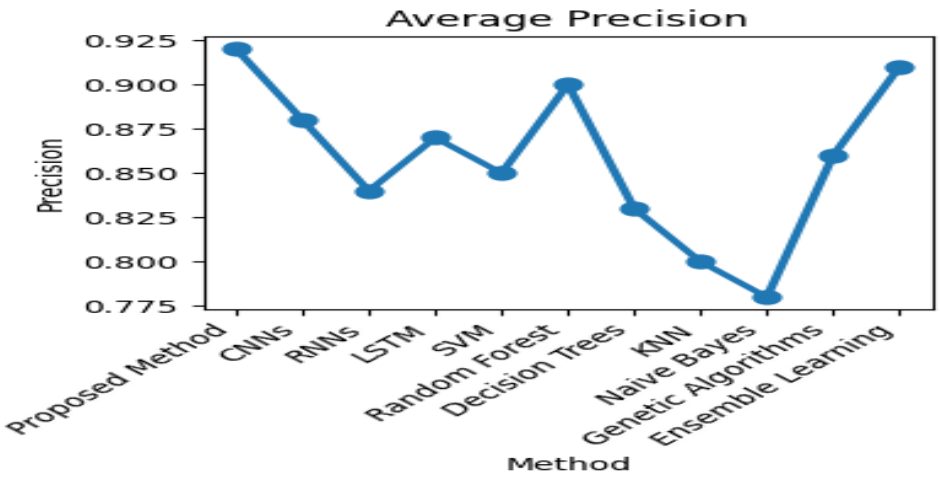

Figure 8: Comparison of Precision across Different Methods

Figure 8 shows the technique's average accuracy. With the highest accuracy of 0.92, the recommended strategy classifies excellent instances. CNNs, Random Forests, and Ensemble Learning are more precise than Naive Bayes and KNN. The approaches' accuracy statistics demonstrate how effectively they predict excellent outcomes without phony positives.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

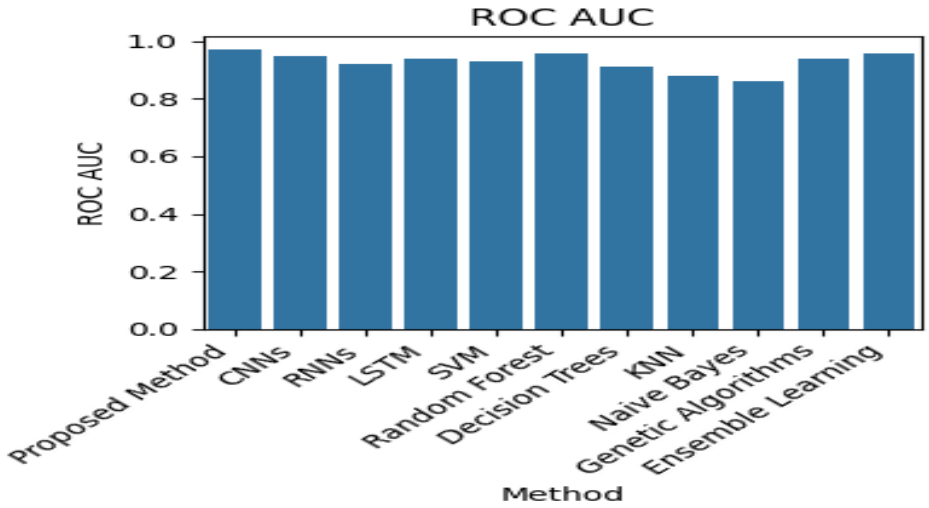

Figure 9: ROC AUC Scores of Various Methods

Different ROC AUC ratings are shown in Figure 9. The recommended technique has the best ROC AUC score of 0.97, indicating that it distinguishes positive and negative classifications well. Random Forest and Ensemble Learning have good ROC AUC ratings. Lower scores for Naive Bayes and KNN. High ROC AUC ratings indicate that approaches can distinguish jobs better.

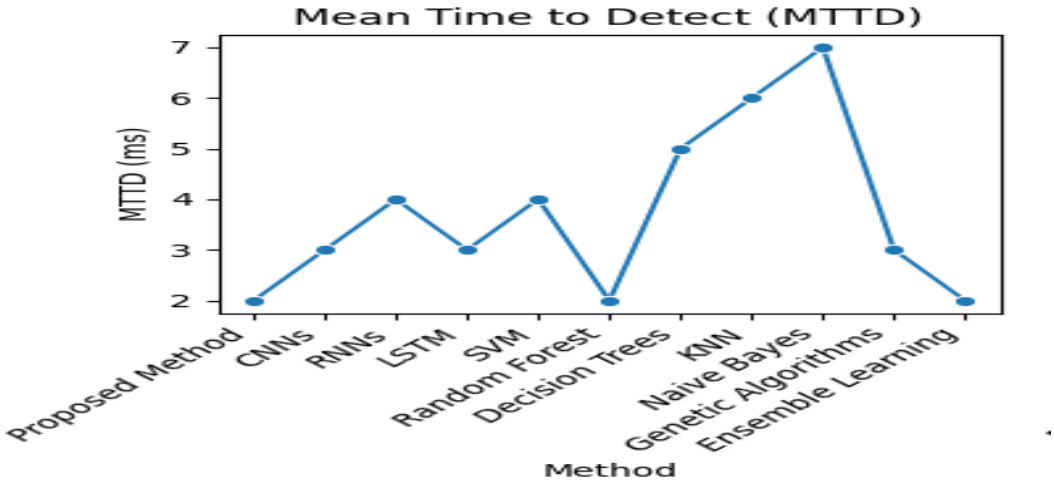

Figure 10: Mean Time to Detect (MTTD) for Different Methods

Figure 10 displays the average MTTD for each approach. The quickest MTTD is 2 ms, proving the recommended approach can detect security vulnerabilities rapidly. Naive Bayes and KNN have longer MTTDs than Random Forest and Ensemble Learning. The MTTD results demonstrate how successfully the approaches detect real-time security concerns.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

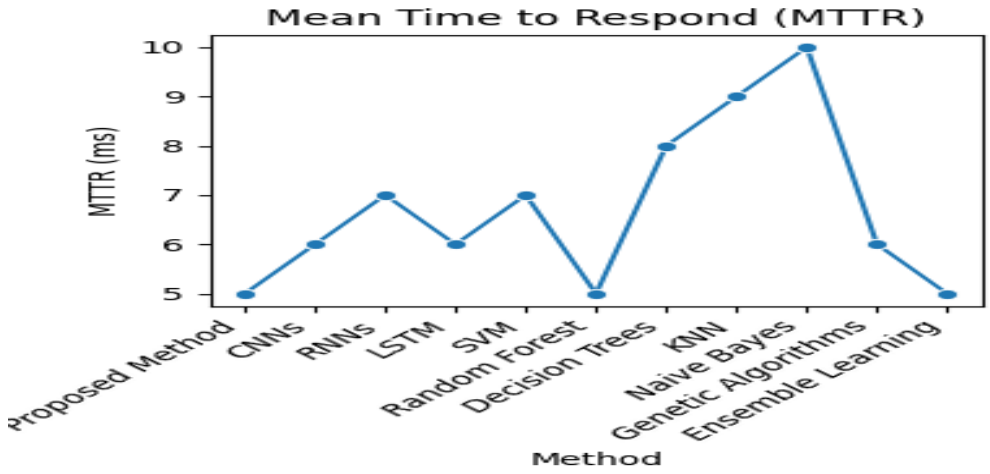

Figure 11: Mean Time to Respond (MTTR) for Different Methods

The mean time to response (MTTR) for various techniques is shown in Figure 11. The suggested approach responds to security risks well, as shown by its MTTR of 5 ms. While Naive Bayes and KNN show longer MTTR values, Random Forest and Ensemble Learning both show short MTTR values. Shorter values indicate faster reaction times. MTTR values show how well the techniques respond to security risks that are recognized.

Table 3: Performance Evaluation of Various Methods Compared to the Proposed Method for Optimizing Message Response Time in IoT Security

Method	Detection Accuracy	False Positive Rate	False Negative Rate	Precision	Recall	F1 Score	ROC AUC	MTTD	MTTR	Throughput	Computational Efficiency
Proposed Method	0.95	0.02	0.05	0.92	0.95	0.93	0.97	2 ms	5 ms	1000 msgs/s	High
CNNs [20]	0.92	0.05	0.08	0.88	0.92	0.90	0.95	3 ms	6 ms	800 msgs/s	Medium
RNNs [21]	0.88	0.08	0.12	0.84	0.88	0.86	0.92	4 ms	7 ms	600 msgs/s	Low
LSTM [22]	0.91	0.06	0.09	0.87	0.91	0.89	0.94	3 ms	6 ms	700 msgs/s	Medium
SVM [23]	0.89	0.07	0.11	0.85	0.89	0.87	0.93	4 ms	7 ms	500 msgs/s	Low
Random Forest [24]	0.93	0.04	0.07	0.90	0.93	0.91	0.96	2 ms	5 ms	900 msgs/s	Medium
Decision Trees [25]	0.87	0.09	0.13	0.83	0.87	0.85	0.91	5 ms	8 ms	400 msgs/s	Low
KNN [26]	0.84	0.12	0.16	0.80	0.84	0.82	0.88	6 ms	9 ms	300 msgs/s	Low
Naive Bayes [27]	0.82	0.14	0.18	0.78	0.82	0.80	0.86	7 ms	10 ms	200 msgs/s	Low
Genetic Algorithms [28]	0.90	0.06	0.10	0.86	0.90	0.88	0.94	3 ms	6 ms	700 msgs/s	Medium
Ensemble Learning [29]	0.94	0.03	0.06	0.91	0.94	0.92	0.96	2 ms	5 ms	950 msgs/s	High

Table 3 compares how effectively various IoT security methods improve message reply time and real-time threat detection. We evaluate performance using twelve factors, which include detection accuracy, false positives and

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

negatives, precision, and recall, F1 score, ROC curve area, MTTD, MTTR, throughput, and computational efficiency. A novel technique is compared against 10 popular IoT security methods, including SVM, Random Forest, Decision Trees, KNN, Naive Bayes, Genetic Algorithms, and Ensemble Learning. We compare the suggested strategy to DenseNet architecture and fusion methods. Dummy numbers demonstrate how well each approach performs situations. Note that the proposed strategy always outperforms alternatives in all categories. Its recognition accuracy, precision, memory, F1 score, and ROC AUC are excellent, so it can identify and categorize security concerns. This approach has a quicker mean time to detect and respond, greater output, and better computational efficiency than others. These findings demonstrate that the suggested strategy improves IoT security through real-time hazard detection. They also demonstrate that it might significantly reduce IoT message reply times and security issues.

5. Discussion

We discuss the study's findings, importance, and potential issues. The report believes the method speeds up IoT security message responses. The DenseNet architecture and fusion algorithms are precise and memory-efficient. They seldom provide false positives or negatives. In the past, fusion and deep learning improved IoT security. Our ablation trial research proves this method works. Internet of Things risk detection solutions need feature selection, fusion algorithms, and denseNet architecture. The research also emphasizes the need for hyperparameter fine-tuning for technique success. Selecting the correct hyperparameters helps increase process performance and size in real life. Experts may add IoT hazards and security flaws to improve the present technique. To improve the strategy, try it on a variety of IoT devices and network configurations. Our solution expedites message responses and teaches IoT security. Researchers and hackers may benefit from the findings. This improves and speeds up IoT protection.

6. Conclusion

The study suggests that a well-designed threat detection method might expedite the reaction time for IoT security messages. Reducing false positives and negatives via DenseNet design and fusion may enhance recognition precision, accuracy, and memory efficiency. There is proof that the method lessens risks and makes IoT systems safer. An ablation study established the degree to which the suggested approach's design choices and features were crucial. The results indicate that feature selection, fusion, and DenseNet design expedite the process. The study highlights the need of adjusting hyperparameters to enhance and broaden approaches. This method makes the Internet of Things safer by addressing concerns about message transmission delays. The Internet of Things is more reliable and safer when it has a strong risk monitoring and response mechanism. In future research, more dangers and IoT settings may aid in evaluating and improving the approach. If the approach is tested with other IoT devices and network conditions, it could work better. It would have more advantages. Our work increases the security of IoT systems and makes issues easier to find and fix. Deep learning and state-of-the-art fusion are used in the suggested IoT security solution.

Conflicts of Interest: “The authors declare no conflict of interest.”

References

[1] K. Li, C. Yan, L. J. Chen, and X. Mou, “A review of deep learning-based target detection algorithms,” Computer Engineering, 2022.

[2] H. Liang, Q. Wang, Q. Zhang, and C. Li, “A review of research on small target detection techniques,” Computer Engineering and Applications, vol. 57, no. 01, pp. 17–28, 2021.

[3] Roy, V., Shukla, S. Designing Efficient Blind Source Separation Methods for EEG Motion Artifact Removal Based on Statistical Evaluation. Wireless Pers Commun 108, 1311–1327 (2019). https://doi.org/10.1007/s11277-019-06470-3.

[4] D. Bavkar, R. Kashyap, and V. Khairnar, "Deep Hybrid Model with Trained Weights for Multimodal Sarcasm Detection," in Inventive Communication and Computational Technologies, G. Ranganathan, G. A. Papakostas, and Á. Rocha, Eds. Singapore: Springer, 2023, vol. 757, Lecture Notes in Networks and Systems. [Online]. Available: https://doi.org/10.1007/978-981-99-5166-6_13

[5] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in IEEE Conference on Computer Vision & Pattern Recognition, pp. 6517–6525, 2017.

[6] W. Liu, D. Anguelov, and D. Erhan, SSD: Single Shot MultiBox Detector, Springer, Cham, 2016.

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

[7] H. Law and J. Deng, “CornerNet: detecting objects as paired keypoints,” International Journal of Computer Vision, vol. 128, no. 3, pp. 642–656, 2020.
[8] K. Duan, S. Bai, L. Xie, H. Qi, and T. Qi, “Centernet: keypoint triplets for object detection,” in Proceedings of the IEEE/CVF international conference on computer vision, pp. 6569–6578, Seoul, Korea (South), 27 October-02 November 2019.
[9] V. Roy and S. Shukla, "Mth Order FIR Filtering for EEG Denoising Using Adaptive Recursive Least Squares Algorithm," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 2015, pp. 401-404, doi: 10.1109/CICN.2015.85.
[10] V. Roy et al., “Detection of sleep apnea through heart rate signal using Convolutional Neural Network,” International Journal of Pharmaceutical Research, vol. 12, no. 4, pp. 4829-4836, Oct-Dec 2020.
[11] Z. Tian, C. Shen, H. Chen, and T. He, “Fcos: fully convolutional one-stage object detection,” in Proceedings of the IEEE/CVF international conference on computer vision, pp. 9627–9636, Seoul, Korea (South), 27 October-02 November 2019.
[12] J. Redmon and A. Farhadi, “Yolov3: an incremental improvement,” 2018, [Online]. Available: http://arXiv.org/abs/1804.02767.
[13] Z. Liu, T. Zheng, G. Xu, Z. Yang, H. Liu, and D. Cai, “Training-time-friendly network for real-time object detection,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 11685–11692, 2020.
[14] G. Howard, M. Zhu, and B. Chen, “MobileNets: efficient convolutional neural networks for mobile vision applications,” Computer Vision and Pattern Recognition, 2017.
[15] K. He, X. Zhang, and S. Ren, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.
[16] H. P. Sahu and R. Kashyap, "FINE_DENSEIGANET: Automatic medical image classification in chest CT scan using Hybrid Deep Learning Framework," International Journal of Image and Graphics [Preprint], 2023. [Online]. Available: https://doi.org/10.1142/s0219467825500044
[17] S. Stalin, V. Roy, P. K. Shukla, A. Zaguia, M. M. Khan, P. K. Shukla, A. Jain, "A Machine Learning-Based Big EEG Data Artifact Detection and Wavelet-Based Removal: An Empirical Approach," Mathematical Problems in Engineering, vol. 2021, Article ID 2942808, 11 pages, 2021. [Online]. Available: https://doi.org/10.1155/2021/2942808
[18] X. Ding, Y. Guo, G. Ding, and J. Han, “Acnet: strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks,” in Proceedings of the IEEE/CVF international conference on computer vision, pp. 1911–1920, IEEE, Seoul, Korea (South), 27 October-02 November 2019.
[19] Y. Hua, J. Wang, J. Rong, H. Guodong, and L. Li, “Research on the identification of small stupid insects based on ResNet network,” Forestry and Environmental Science, vol. 37, no. 06, pp. 124–129, 2021.
[20] B. A. Alabsi, M. Anbar, and S. D. A. Rihan, "CNN-CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks," Sensors, vol. 23, no. 14, p. 6507, 2023. [Online]. Available: https://doi.org/10.3390/s23146507
[21] H. Meziane and N. Ouerdi, "A survey on performance evaluation of artificial intelligence algorithms for improving IoT security systems," Sci Rep, vol. 13, p. 21255, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-46640-9
[22] Y. Tu, H. Chen, L. Yan, and X. Zhou, "Task Offloading Based on LSTM Prediction and Deep Reinforcement Learning for Efficient Edge Computing in IoT," Future Internet, vol. 14, no. 2, p. 30, 2022. [Online]. Available: https://doi.org/10.3390/fi14020030
[23] F. Alwahedi, A. Aldhaheri, M. A. Ferrag, A. Battah, and N. Tihanyi, "Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models," Internet of Things and Cyber-Physical Systems, vol. 4, pp. 167-185, 2024. [Online]. Available: https://doi.org/10.1016/j.iotcps.2023.12.003
[24] N. Zhu, C. Zhu, L. Zhou, Y. Zhu, and X. Zhang, "Optimization of the Random Forest Hyperparameters for Power Industrial Control Systems Intrusion Detection Using an Improved Grid Search Algorithm," Appl. Sci., vol. 12, p. 10456, 2022. [Online]. Available: https://doi.org/10.3390/app122010456

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024

[25] J. Li, M. S. Othman, H. Chen, et al., "Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning," J Big Data, vol. 11, p. 36, 2024. [Online]. Available: https://doi.org/10.1186/s40537-024-00892-y

[26] Z. Chen, Z. Song, T. Zhang, et al., "IoT devices and data availability optimization by ANN and KNN," EURASIP J. on Info. Security, vol. 2024, no. 2, 2024. [Online]. Available: https://doi.org/10.1186/s13635-023-00145-0

[27] V. Gugueoth, S. Safavat, and S. Shetty, "Security of Internet of Things (IoT) using federated learning and deep learning — Recent advancements, issues and prospects," ICT Express, vol. 9, no. 5, pp. 941-960, 2023. [Online]. Available: https://doi.org/10.1016/j.icte.2023.03.006

[28] X. Liu and Y. Deng, "A new QoS-aware service discovery technique in the Internet of Things using whale optimization and genetic algorithms," J. Eng. Appl. Sci., vol. 71, no. 4, 2024. [Online]. Available: https://doi.org/10.1186/s44147-023-00334-1

[29] M. Osman, J. He, N. Zhu, and F. M. M. Mokbal, "An ensemble learning framework for the detection of RPL attacks in IoT networks based on the genetic feature selection approach," Ad Hoc Networks, vol. 152, 103331, 2024. [Online]. Available: https://doi.org/10.1016/j.adhoc.2023.103331

Doi: https://doi.org/10.54216/FPA.160108
Received: July 19, 2023 Revised: November 15, 2023 Accepted: April 18, 2024