Journal of Information Systems Engineering and Management
2025, 10(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Optimized Deep Feature Analysis for Enhanced Botnet Attack Prediction in IoT Networks

[1]Sudarshan S. Sonawane

[1]*Department of Computer Engineering, R. C. Patel Institute of Technology, Shirpur, India*

[1]*Email: sudars2000@gmail.com*

ARTICLE INFO

Received: 29 Sept 2024

Revised: 30 Nov 2024

Accepted: 12 Dec 2024

ABSTRACT

The Optimized Deep Feature Analysis (ODFA) framework targets botnet attacks in IoT networks by combining deep learning with machine learning techniques. It uses a hybrid CNN-LSTM model to extract spatial and temporal features from network traffic, applying Recursive Feature Elimination (RFE) with Cross-Validation (CV) for refining feature selection. A Random Forest classifier is then employed to classify different types of botnet attacks. Tests conducted on the UNSW-NB15 dataset demonstrate its ability to achieve high precision, sensitivity, specificity, and accuracy. Recurrent feature optimization allows the framework to adapt to changes in attack patterns, ensuring consistent detection performance. The reduction in false alarms and improved classification of attack types highlights its contribution to securing IoT networks.

Keywords: Recursive Feature Elimination, Cross-Validation, Botnet Attack Prediction, Internet of Things, CNN, Deep Auto Encoder.

INTRODUCTION

Internet of Things (IoT) networks face increasing security concerns due to the widespread adoption of smart devices and their extensive connectivity [1]. These networks are widely used in homes, healthcare, industries, and cities, making their security measures crucial [2]. Botnet attacks pose a significant threat to IoT networks, exploiting their complex architecture and continuous data exchange, often causing severe disruptions [3], [4].

Traditional security measures struggle to counter advanced botnet attacks [5]. This challenge calls for adaptable frameworks that can detect and classify such threats effectively [6]. The Optimized Deep Feature Analysis (ODFA) framework addresses this need by combining deep learning and machine learning techniques for botnet attack prediction in IoT networks.

The framework follows a structured approach:

1. Combines CNN and LSTM models to analyze spatial and temporal patterns in network traffic.
2. Applies Recursive Feature Elimination (RFE) with Cross-Validation (CV) to refine and select relevant features.
3. Employs Random Forest for multi-label classification of botnet attacks.
4. Includes recurrent feature updates to adapt to changing attack strategies.

ODFA focuses on identifying patterns in network traffic while adapting to new threats over time. It uses computational techniques to classify and predict botnet attacks in a structured manner, providing a focused response to IoT security challenges.

RELATED RESEARCH

The field of IoT network security has focused on addressing botnet attacks using machine learning and deep learning methods. Researchers have developed various strategies to improve detection and classification.

Soe et al. [7] introduced a sequential architecture model that prioritized accuracy while reducing false positives. Al-Sarem et al. [8] proposed a method combining AMI-based feature selection with machine learning to enhance classifier accuracy. Guerra et al. [9] evaluated feature selection techniques to improve interpretability and performance in IoT botnet detection. Deep learning has been explored extensively. Sriram et al. [10] investigated

deep neural networks for detecting botnets in IoT devices and smart city applications. Popoola et al. [11] developed LS-DRNN, a memory-efficient model integrating multiple techniques for classification. Stiawan et al. [12] used deep autoencoders and neural networks to reduce data dimensionality for effective detection.

Data privacy concerns in IoT networks led to federated deep learning frameworks. Popoola et al. [13, 14] designed such frameworks to detect botnets on IoT-edge devices while preserving privacy. Other studies addressed specific IoT applications. Dharini et al. [15] and Azhari et al. [16] focused on devices like routers and cameras, applying machine learning for detection. Optimization techniques for deep learning in IoT security have also been studied. Kalidindi et al. [17], Sangher et al. [18], and Popoola et al. [19] worked on enhancing accuracy and efficiency in botnet detection models. Negera et al. [20] reviewed machine learning in SDN-enabled IoT networks, while Masoodi et al. [21] compared ensemble learning techniques.

Researchers have also explored detecting specific attack types. Jeyanthi et al. [22] and Alissa et al. [23] applied machine learning for botnet detection. Al-Marsoomi et al. [24] and Batham et al. [25] used deep learning for DDoS and IoT botnet detection. Borra et al. [26] developed machine learning algorithms incorporating login puzzles and federated detection. Karaca et al. [27] focused on convolutional neural networks for network traffic classification.

These studies emphasize the use of techniques like feature selection, federated learning, and deep models to address IoT botnet threats.

ODFA FRAMEWORK

The Optimized Deep Feature Analysis (ODFA) framework focuses on improving IoT network security by detecting and classifying botnet attacks. It combines deep learning and machine learning methods to analyze IoT network traffic and identify threats. The approach integrates several components, including data collection, preprocessing, feature extraction, and training, to address complex cyber threats systematically.

Data Collection and Preprocessing

The foundation for analysis and prediction was laid by the methodical execution of the data collection and preprocessing phase. This crucial phase involved well-planned steps to guarantee the best data quality and relevance for upcoming tasks.

Large-scale data collection on IoT network traffic was the first step in the process. From network activities and botnet attacks, a sizable and varied data set was gathered. This collection comprises IoT data from multiple devices and networks, both simulated and real-world. The depth and realism of the data were improved through partnerships with network administrators and open datasets. The ODFA models' capacity to reliably detect botnet attacks and generalize across scenarios has been greatly aided by this data collection phase.

Normalization and standardization started after data collection. This was a critical step that involved bringing data scales and units into alignment so that the mean was zero and the standard deviation was one. A meticulous process of normalization reduced feature values to 0–1. This step stopped any single feature from controlling the model's learning process because of scale disparities. For data uniformity and consistency, Python's NumPy, Pandas, and Scikit-learn were utilized.

The final step in the preprocessing process was structuring the data into formats suitable for deep learning applications. The network traffic data was arranged using precise time windows and sequences. Sequencing followed the transmission of network packets, whereas timing involved segmenting the data into fixed-size intervals that contained all pertinent data points. This structuring was necessary for deep learning models, such as CNNs and LSTMs, to learn and understand the spatial and temporal patterns of network traffic. The efficacy of ODFA hinges on its capacity to differentiate botnet activity from regular operations.

Feature Extraction with CNN-LSTM

The ODFA framework employs a hybrid model that integrates Convolutional Neural Networks (CNNs) [28] and Long Short-Term Memory networks (LSTMs) [29]. This combination allows the framework to extract both spatial and temporal patterns from network traffic data figure 1. Here’s an outline for developing such a model:

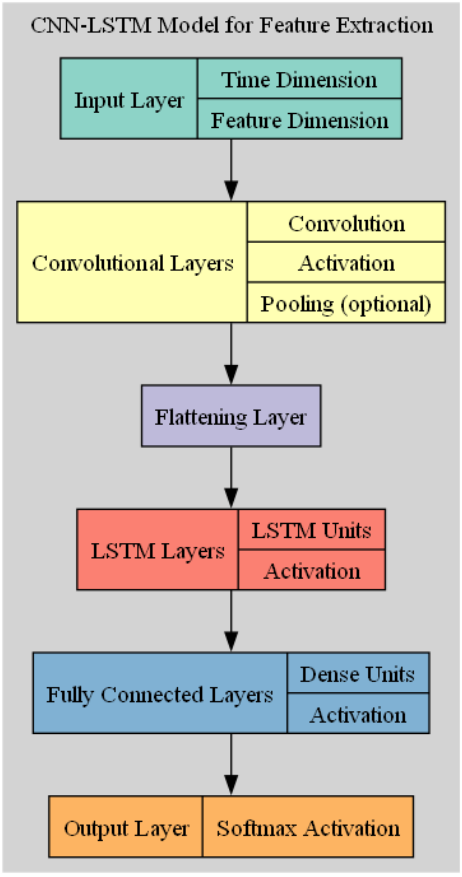

Figure 1: CNN-LSTM architecture

- Input Layer: Accepts preprocessed traffic data, structured as a 2D array with one axis representing time and the other containing network features.
- Convolutional Layers: Apply filters to extract spatial features, highlighting relationships between data points. Pooling layers reduce dimensions to minimize computational complexity and control overfitting.
- Flattening Layer: Converts multi-dimensional outputs from the CNN into a one-dimensional array for processing by LSTM layers.
- LSTM Layers: Capture temporal dependencies and long-term patterns in the network traffic sequences.
- Fully Connected Layers: Combine high-level features extracted by CNNs and LSTMs to learn complex relationships in the data.
- Output Layer: Classifies network traffic, outputting probabilities for different attack types using a softmax activation function.

Training the Model

- **Compilation:** The model uses the Adam optimizer, categorical cross-entropy as the loss function for multi-class classification, and accuracy as the evaluation metric.
- **Training:** The network learns from labeled examples in the dataset, adjusting weights to improve classification accuracy.
- **Validation:** A validation set monitors performance on unseen data to identify issues such as overfitting and guide further adjustments.

The ODFA framework employs this structured process to enhance the detection and classification of botnet attacks in IoT networks, focusing on practical, data-driven solutions.

Feature Optimization with RFE and CV

The feature optimization process in the ODFA framework improves the set of input features used for predicting botnet attacks in IoT networks. It applies Recursive Feature Elimination (RFE) [30], Cross-Validation (CV) [31], and SHAP (SHapley Additive exPlanations) [32] to identify and retain the most relevant features that shown in figure 2. This approach refines the CNN-LSTM model's input, enhancing its accuracy and reducing unnecessary complexity. Below is a breakdown of the feature optimization process:

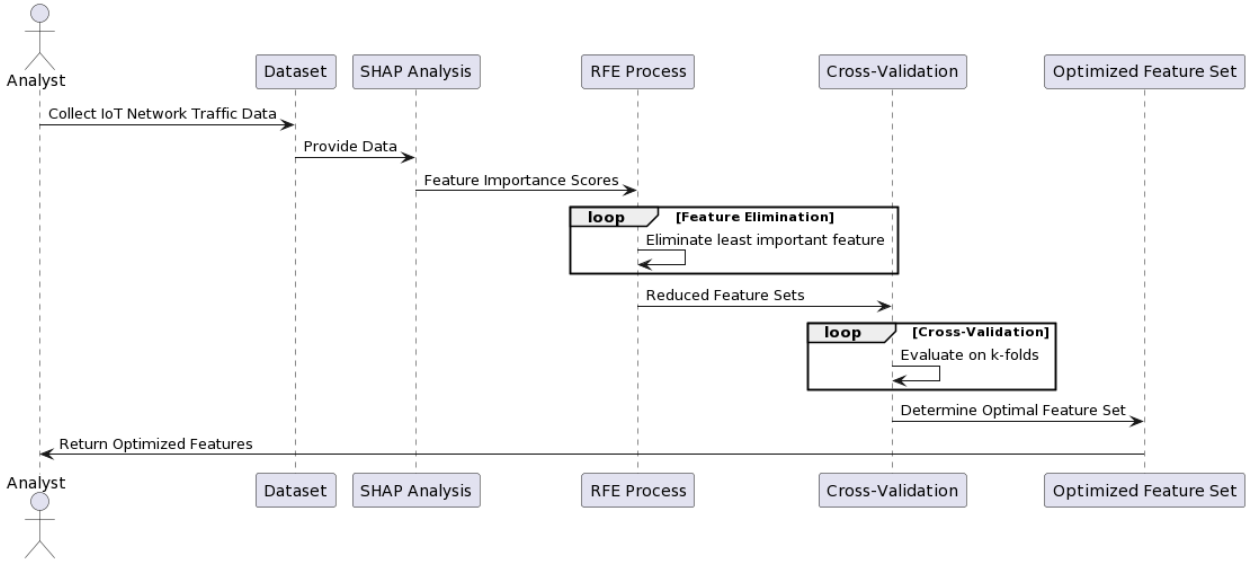

Figure 2: Flow Diagram of RFE and CV

Using SHAP for Feature Evaluation: SHAP is used to evaluate the importance of each feature in the model. By analyzing the CNN-LSTM predictions, SHAP assigns scores that reflect how much each feature contributes to identifying botnet activity. Features with higher scores are prioritized for inclusion in the model, ensuring that only the most informative attributes are retained.

Recursive Feature Elimination (RFE): RFE iteratively removes the least important features from the dataset. Initially, the process starts with all features and eliminates the ones with the lowest importance scores, as determined by SHAP. After each removal, the CNN-LSTM model is retrained on the remaining features. This retraining step is critical because the importance of a feature can change as others are removed. The iterative nature of RFE ensures that the model is refined to use only the most essential attributes.

Integration of Cross-Validation (CV): Cross-validation is incorporated into the RFE process to improve the generalizability of the selected features. The dataset is divided into multiple subsets, and RFE is applied to each subset separately. This method validates the selected features across different portions of the data, minimizing the risk of overfitting. It ensures that the chosen features work well with both the training and unseen datasets.

Identifying the Optimal Feature Set: The combination of RFE and CV identifies the smallest set of features that maintains high model performance. Metrics such as accuracy, precision, recall, and F1-score are tracked throughout the process to evaluate how the removal of features affects classification. The optimization process concludes when the model achieves a balance between the number of features and its predictive performance. Eliminating too many features often leads to reduced accuracy, which signals the stopping point for feature reduction.

This systematic approach ensures that the final model uses an efficient feature set tailored to accurately detect and classify botnet attacks in IoT networks.

- **Algorithm: Feature Optimization in ODFA Framework**

Input:

- D: IoT Network Traffic Data
- M : Pre-trained CNN-LSTM Model
- N : Total number of features in D

Output:

- F_{opt} : Optimized set of features

Procedure:

1. **Initialize**:
   - Set F = All features in D
   - Define PerformanceMetric(·) to evaluate model performance (e.g., accuracy, F1-score)
2. **Apply SHAP for Feature Importance**:
   - Use SHAP on M with dataset D to get feature importances:
   - Importance Scores=SHAP$\left(M, D\right)$
3. **Feature Elimination Process**:
   - **For** $i = 1$ **to** N **do**:
     - Sort F based on Importance Scores in ascending order
     - Remove the least important feature from F
     - $F = F -$least important feature}
4. **Cross-Validation with RFE**:
   - Divide D into k folds for CV
   - **For each** fold in D **do**:
     - **For** each reduced feature set F_i **do**:
       - Train M on $k-1$folds with F_i
       - Evaluate M on the remaining fold
       - Calculate PerformanceMetric$\left(M\right)$
     - Record average performance across all folds
5. **Identify Optimal Feature Set**:
   - Analyze the recorded performances for each feature set
   - Identify the set F_{opt} that yields the best balance between model performance and feature minimization
6. **Retrain Model on** F_{opt} :
   - Retrain M on the entire dataset D using only features in F_{opt}

Return: F_{opt}

- **Explanation**
- This algorithm systematically reduces the number of features by eliminating the least important ones based on SHAP scores.
- The integration of cross-validation ensures that the feature selection is robust and generalizable across different subsets of the data.
- The process aims to find an optimized set of features F_{opt} that maintains high model performance while reducing complexity and computational load.
- The final step involves retraining the CNN-LSTM model with the optimized feature set to ensure it is fine-tuned for the best possible performance on the reduced feature set.

Botnet Attack Classification with Random Forest

The Botnet Attack Classification stage in the ODFA framework uses a Random Forest [33] classifier to identify various types of botnet attacks. This phase relies on the optimized features selected through Recursive Feature Elimination (RFE) and Cross-Validation (CV), ensuring only the most relevant attributes are used for training.

Training the Random Forest Classifier

The classification process begins with the optimal feature set identified in the feature optimization phase. Random Forest, an ensemble learning method, constructs multiple decision trees using this feature set. Each tree is trained on a random subset of features and samples, allowing the model to make predictions by combining the outputs of these trees.

The classifier is designed for multi-label classification, enabling it to differentiate between multiple types of botnet attacks simultaneously. By splitting the dataset into smaller subsets, each decision tree in the forest learns unique patterns, enhancing its ability to classify diverse attack types accurately.

Fine-Tuning Model Parameters

Random Forest performance depends on several key parameters that require careful adjustment:

Number of Trees (n_estimators): This parameter determines how many decision trees are included in the forest. Increasing the number of trees often improves accuracy but also raises computational costs. A balance is maintained to optimize both performance and efficiency.

Depth of Trees (max_depth): The depth of each tree influences how well the model can capture patterns in the data. Shallow trees may miss complex relationships, while deeper trees risk overfitting. Adjusting this parameter ensures the model captures sufficient detail without becoming overly specific.

Additional Parameters: Parameters like min_samples_split, min_samples_leaf, and max_features control how trees split data and select features. Fine-tuning these values ensures that the trees are neither too shallow nor too complex, enhancing the classifier's overall performance.

Fine-tuning these parameters ensures the Random Forest classifier achieves a balance between computational efficiency and prediction accuracy. Adjustments are guided by performance metrics such as precision, recall, and accuracy, which indicate how well the model distinguishes between normal and malicious traffic.

This classification phase integrates the optimized feature set and parameter adjustments to train a model that identifies botnet attacks effectively. By leveraging the strengths of Random Forest, this stage contributes to the comprehensive detection capabilities of the ODFA framework.

- **Algorithm: Botnet Attack Classification with Random Forest**

Inputs:

- F_{opt} : Optimized feature set from RFE and CV
- S_{train} : Training dataset with input-output pairs (X, y)
- S_{valid} : Validation dataset with input-output pairs (X_{valid}, y_{valid})

Hyperparameters:

- n : Number of trees in the forest
- d : Maximum depth of each tree
- m : Minimum samples required to split a node
- l : Minimum samples required at a leaf node

Output:

- MRF : Trained Random Forest model
- Performance Metrics: Evaluation metrics such as accuracy, precision, recall, and F1-score

Algorithm Steps:

1. **Initialize Random Forest Classifier**:

- Instantiate Random Forest model MRF with hyperparameters $n, d, m, and\ l$.

2. **Train Random Forest**:
   - Train MRF using F_{opt} and S_{train} .
   - For each tree in MRF , randomly select features from F_{opt} and data points from S_{train} to fit the tree.
3. **Hyperparameter Tuning**:
   - Define a grid of hyperparameters to search over.
   - Perform grid search with cross-validation on S_{train} to find the optimal hyperparameters $n_{opt}, d_{opt}, m_{opt}, and\ l_{opt}$.
   - Update MRF with optimal hyperparameters.
4. **Model Validation**:
   - Validate the updated MRF on S_{valid} .
   - Calculate PerformanceMetrics on S_{valid} to assess the model's prediction accuracy.
5. **Performance Evaluation**:
   - If PerformanceMetrics are satisfactory, proceed to deployment.
   - If not, revisit hyperparameter tuning or feature set optimization.
6. **Return Model and Metrics**:
   - Return the trained model MRF and PerformanceMetrics.

Return MRF , PerformanceMetrics

Recurrent Feature Optimization

The recurrent feature optimization phase in the ODFA framework shown in figure 3 continuously updates the feature set to ensure it aligns with changing network conditions and botnet attack strategies. This phase maintains the classification system's reliability by adapting to evolving data patterns. An in-depth explanation of this phase:

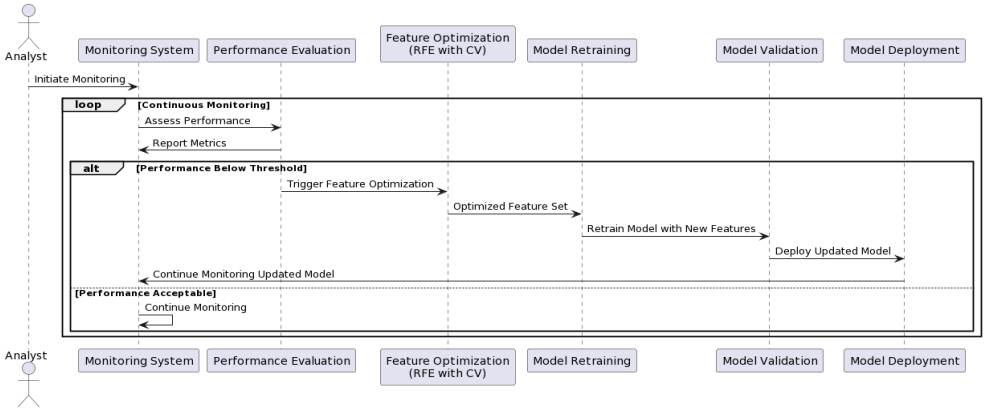

Figure 3: Data Flow in Recurrent Feature Optimization

The system regularly evaluates classification performance, focusing on Type I (false positive) and Type II (false negative) errors. A monitoring mechanism tracks predictions and compares them to actual outcomes. Metrics such as recall and precision are used to assess and improve the model's ability to reduce false classifications.

IoT networks often experience shifts in traffic patterns and security risks, including the emergence of new attack types. The monitoring system identifies these shifts by detecting changes in data patterns or performance metrics. A decrease in classification accuracy may indicate that the feature set is no longer capturing critical characteristics of the data.

When performance changes are detected, the framework applies Recursive Feature Elimination (RFE) and Cross-Validation (CV) to update the feature set. Using the latest data, RFE removes features that contribute less to classification accuracy. Cross-validation ensures that the updated features are consistent and generalizable across different data subsets.

After the feature set is updated, the Random Forest classifier is retrained to incorporate the new features. This step ensures that the model uses the most relevant attributes to improve prediction accuracy. Retraining with refined features allows the system to stay effective against both known and newly observed attack patterns.

The recurrent optimization process ensures that the model adapts to evolving attack strategies and dynamic network environments. By continually updating the feature set and retraining the model, the framework maintains its ability to classify botnet attacks accurately. This iterative approach provides a structured method to address ongoing changes in IoT traffic and security challenges.

- **Algorithm: Recurrent Feature Optimization in ODFA Framework**

Inputs:

- M : Trained Random Forest Model for Botnet Attack Classification
- D : Continuous stream of IoT Network Traffic Data
- F : Current set of features used in M
- T : Threshold for performance metrics (e.g., F1-score, precision, recall)

Outputs:

- $F_{updated}$: Updated set of features
- $M_{updated}$: Updated Random Forest Model

Procedure:

Initialize Monitoring System:

- Set up a system to continuously monitor M 's performance on new data D .

Continuous Performance Evaluation:

- At regular intervals or after a certain number of predictions, evaluate M on the latest data.
- Calculate performance metrics: accuracy, precision, recall, F1-score, etc.

Check Performance Against Threshold:

- **If** performance metrics fall below threshold T **then**:
  - Trigger feature optimization and model update process.
- **Else**:
  - Continue monitoring.

Feature Optimization (RFE with CV):

- **If** feature optimization is triggered **then**:
  - Use SHAP or a similar approach to evaluate the importance of features in F based on the latest data.
  - Apply Recursive Feature Elimination (RFE) combined with Cross-Validation (CV) on D using the current M to identify the new optimized feature set $F_{updated}$.

Update the Classification Model:

- Retrain M using $F_{updated}$ to obtain $M_{updated}$.
- Validate $M_{updated}$ on a separate validation set.

Deploy Updated Model:

- Deploy $M_{updated}$ for real-time classification tasks.
- Update F and M to $F_{updated}$ and $M_{updated}$ respectively.

Return to Monitoring:

- Continue monitoring $M_{updated}$ performance on ongoing traffic data D.

Return: $F_{updated}, M_{updated}$

EXPERIMENTAL STUDY

This section evaluated the Optimized Deep Feature Analysis (ODFA) framework using the UNSW-NB15 dataset [34], which includes diverse network activities such as worms, exploits, denial-of-service (DoS) attacks, and fuzzers. Both benign and malicious traffic patterns were represented in the dataset. Preprocessing steps were applied to ensure compatibility with the ODFA framework. These included normalization, standardization, and structuring of data to meet the model's input requirements. The study applied a four-fold cross-validation approach to assess the framework's reliability. The dataset was divided into four equal subsets, with three subsets used for training and one for validation in each fold. This rotation ensured that all subsets were used for both training and validation, improving the robustness of the results. ODFA's performance was compared with Deep Auto Encoder (DAE) [12] and Deep Neural Network (DNN) [19] models to evaluate its ability to identify and categorize network attacks. The experiments were conducted using high-performance computing resources, including a multicore CPU, an advanced GPU for deep learning tasks, and high-capacity RAM. Python programming was used for implementation, leveraging libraries such as Pandas and NumPy for data handling and Scikit-learn for model training and evaluation [35].

Performance Analysis

The performance analysis utilized the UNSW-NB15 dataset and four-fold cross-validation to measure ODFA's ability to detect and classify botnet attacks. Precision, recall, accuracy, and other metrics were calculated to provide a detailed understanding of the framework's effectiveness. The results demonstrated its ability to distinguish between attack and normal traffic patterns with a consistent and measurable improvement over alternative methods, such as DAE and DNN. Each metric highlighted how the model responded to varying types of network activity, ensuring that both false positives and false negatives were minimized. The detailed analysis confirmed that the framework adapts well to different attack scenarios, showcasing its suitability for real-world IoT security applications.

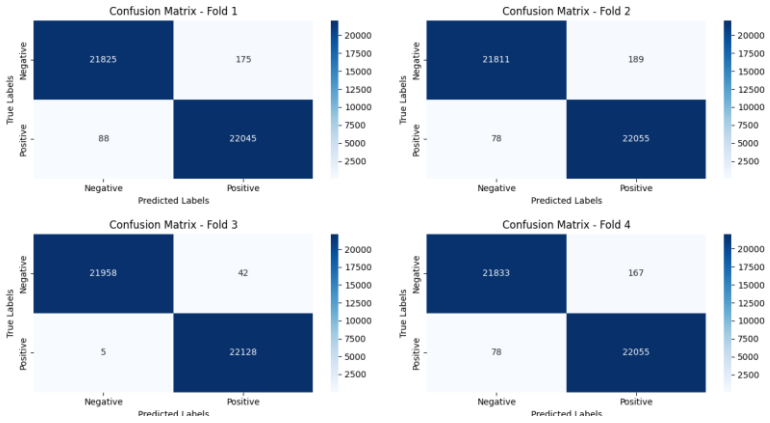

Figure 4: Confusion Matrices of four fold cross validation performed on ODFA

The confusion matrices in figure 4 show the performance metrics for each fold during the evaluation of the ODFA framework. In the first fold, ODFA achieved a precision of 0.9921, highlighting its accuracy in identifying botnet attacks while maintaining minimal false alarms. The sensitivity rate was 0.996039, with 88 false negatives, indicating that most actual attacks were correctly identified. Specificity reached 0.992055, reflecting its ability to correctly recognize non-attack instances.

The second fold table 1 maintained similar levels of precision, sensitivity, and specificity, with a slight increase in false positives. The third fold displayed the highest precision and sensitivity, ensuring near-perfect detection of botnet

attacks. The fourth fold showed a slight decline compared to the third but retained consistent performance across all metrics.

Table 1: Statistics of fourfold cross validation observed from ODFA

ODFA							
	Precision	specificity	sensitivity	accuracy	f-measure	false alarming	MCC
Fold#1	0.992	0.992	0.996	0.994	0.992	0.006	0.988
Fold#2	0.992	0.991	0.996	0.994	0.992	0.006	0.988
Fold#3	0.998	0.998	1	0.999	0.998	0.001	0.998
Fold#4	0.993	0.992	0.996	0.994	0.992	0.006	0.989

The performance analysis of the Deep Auto Encoder (DAE) based approach, also conducted through a four-fold cross-validation, revealed a consistent and commendable level of effectiveness in IoT network security.

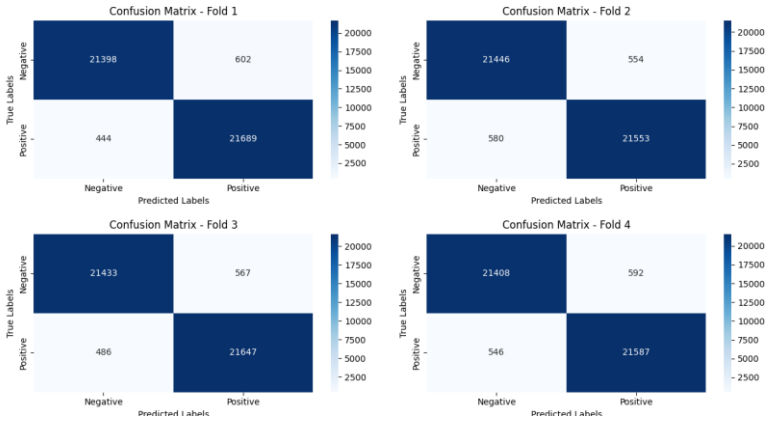

Figure 5: Confusion Matrices of four fold cross validation performed on DAE

The DAE model, as shown in figure 5 and table 2, achieved precision rates between 0.973 and 0.9749 across all folds. Specificity hovered just above 0.97, and sensitivity remained near or above 0.975, indicating reliable detection of positive instances with relatively low false positives.

Table 2: Statistics of fourfold cross validation observed from DAE

DAE							
	precision	specificity	sensitivity	accuracy	fmeasure	false alarming	MCC
Fold#1	0.986	0.986	0.963	0.974	0.986	0.026	0.949
Fold#2	0.989	0.989	0.966	0.978	0.989	0.022	0.956
Fold#3	0.984	0.984	0.969	0.976	0.984	0.024	0.953
Fold#4	0.985	0.985	0.961	0.973	0.985	0.027	0.947

The performance analysis of the Deep Neural Network (DNN) based approach provided substantial insights into its strengths and areas for improvement in IoT network security.

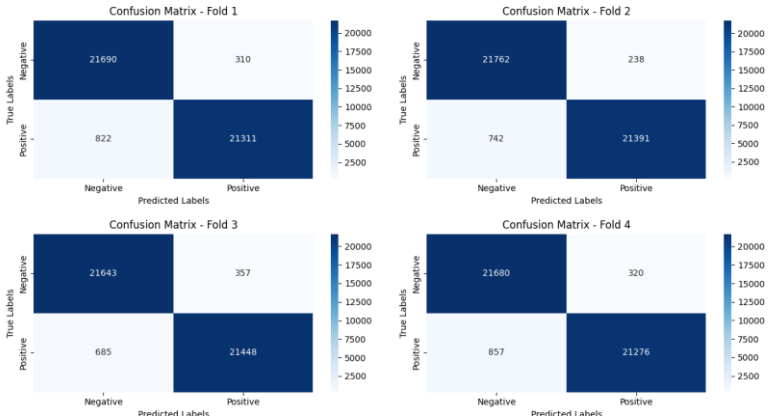

Figure 6: Confusion Matrices of four fold cross validation performed on DNN

The DNN model, described in figure 6 and table 3, showed precision rates ranging from 0.9857 to 0.989 across all folds. Specificity was slightly above 0.98, and sensitivity values were close to or exceeded 0.96. This suggests the model could distinguish between normal and attack traffic effectively, though it occasionally misclassified instances.

Table 3: Statistics of fourfold cross validation observed from DNN

DNN							
	Precision	specificity	sensitivity	accuracy	f-measure	false alarming	MCC
Fold#1	0.973	0.973	0.98	0.976	0.973	0.024	0.953
Fold#2	0.975	0.975	0.974	0.974	0.975	0.026	0.949
Fold#3	0.975	0.974	0.978	0.976	0.974	0.024	0.952
Fold#4	0.973	0.973	0.975	0.974	0.973	0.026	0.948

Comparative Analysis

The comparative analysis revealed that the ODFA framework consistently outperformed both the DAE and DNN approaches across various performance metrics.

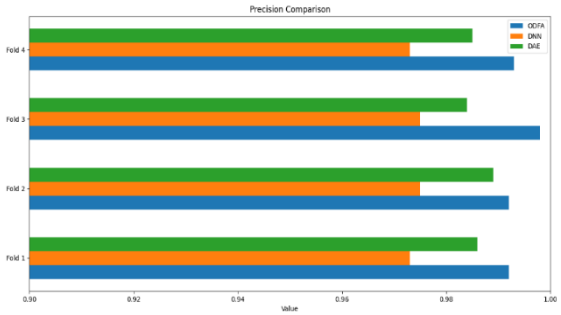

Figure 7: Comparison of precision statistics observed for OFDA, ADE, and DNN

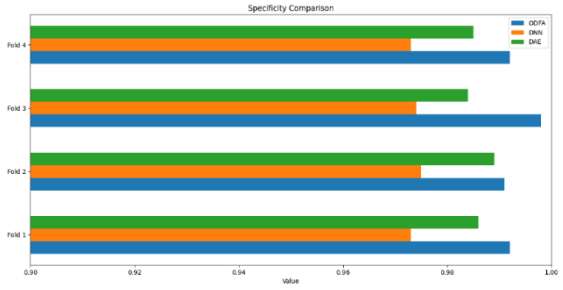

Figure 8: Comparison of specificity statistics observed for OFDA, ADE, and DNN

Precision metrics in figure 7 demonstrated that ODFA reduced false positives more effectively than DAE and DNN. Figure 8 showed ODFA achieving the highest specificity, consistently identifying non-attack instances accurately.

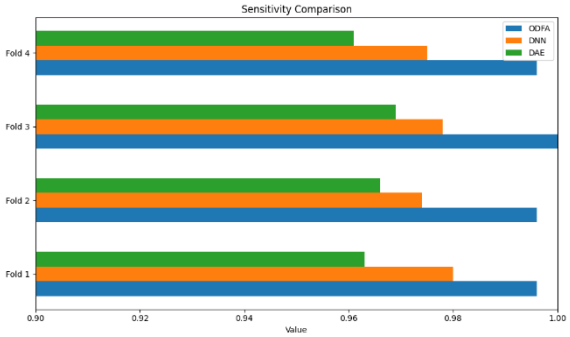

Figure 9: Comparison of sensitivity statistics observed for OFDA, ADE, and DNN

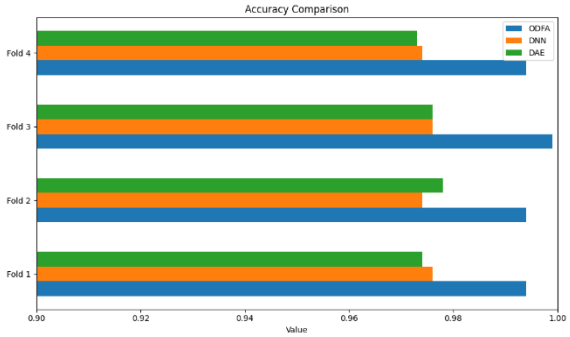

Figure 10: Comparison of Accuracy statistics observed for OFDA, ADE, and DNN

Sensitivity metrics in figure 9 indicated that ODFA detected nearly all true positives, surpassing DNN, which performed well but remained slightly behind. DAE showed the lowest sensitivity among the three models. Accuracy comparisons in figure 10 confirmed ODFA's ability to classify both attack and non-attack instances with higher reliability than DAE and DNN.

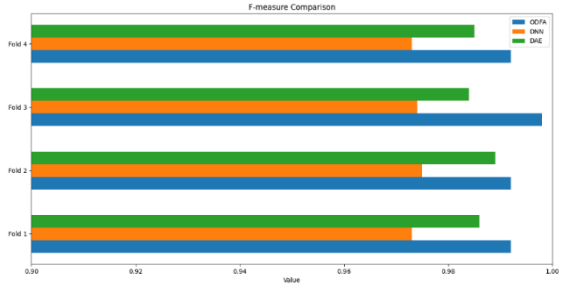

Figure 11: Comparison of F-measure statistics observed for OFDA, ADE, and DNN

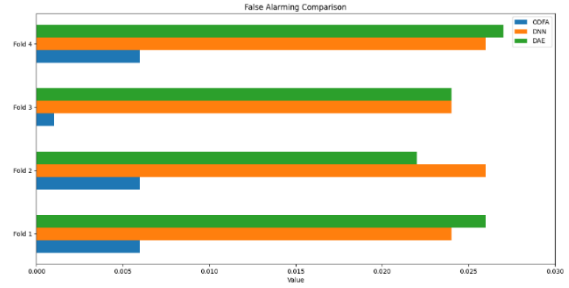

Figure 12: Comparison of false alarming statistics observed for OFDA, ADE, and DNN

Figure 11 presented the F-measure, which balances precision and recall, showing ODFA with the highest scores across all folds. False alarm rates in figure 12 were lowest for ODFA, reducing unnecessary responses to non-attacks. DNN and DAE exhibited higher false alarm rates, with DNN recording the most false positives.

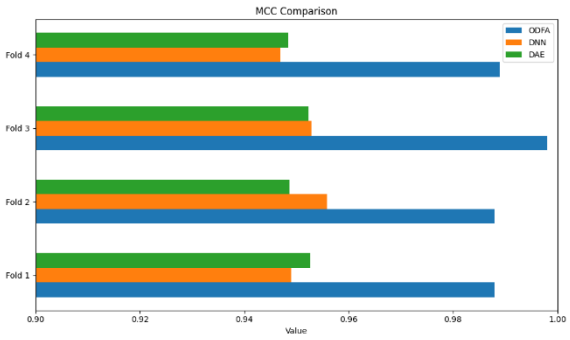

Figure 13: Comparison of MCC statistics observed for OFDA, ADE, and DNN

Finally, figure 13 displayed the Matthews Correlation Coefficient (MCC), where ODFA achieved the highest scores, reflecting strong agreement between predicted and actual classifications.

The performance and comparative analyses highlight the strengths of the ODFA framework for detecting and classifying botnet attacks in IoT networks. It demonstrated consistent performance across metrics such as precision, specificity, sensitivity, accuracy, F-measure, false alarm rate, and MCC. These results underline its suitability for addressing the challenges of IoT security, ensuring reliable classification under diverse conditions.

CONCLUSION

The Optimized Deep Feature Analysis (ODFA) framework provides a structured approach to improving IoT network security by detecting and classifying botnet attacks. Through experimental evaluation, ODFA demonstrated its ability to accurately identify and categorize various types of attacks while minimizing false alarms. By integrating deep

learning and machine learning techniques, the framework improves classification accuracy compared to traditional approaches. A key component of ODFA is its adaptive feature optimization process, which uses Recursive Feature Elimination (RFE) with Cross-Validation (CV) to refine the feature set continually. This process allows the framework to adjust to changes in network behavior and attack patterns. The hybrid model, which combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), captures spatial and temporal patterns in IoT network traffic, improving the model's ability to process complex data. The experimental results consistently showed ODFA outperforming alternative methods, such as Deep Auto Encoder (DAE) and Deep Neural Network (DNN), across metrics like precision, specificity, sensitivity, accuracy, false alarm rate, and Matthews Correlation Coefficient (MCC). These outcomes highlight its ability to handle dynamic and complex traffic in IoT networks. ODFA's recurrent feature optimization ensures continued effectiveness by updating its feature set as new attack patterns emerge. This iterative approach helps the model remain relevant in the changing landscape of IoT security threats. The framework balances accurate detection with a low false alarm rate, making it suitable for real-world security applications. The ODFA framework addresses the challenges of botnet detection in IoT environments by leveraging data-driven techniques. As IoT networks grow and become more interconnected, frameworks like ODFA provide practical tools to enhance the security and reliability of these systems. This approach offers a scalable solution to mitigate evolving cyber threats.

REFERENCES

[1]. Bhattacharjya, Aniruddha, Xiaofeng Zhong, Jing Wang, and Xing Li. "Security challenges and concerns of Internet of Things (IoT)." Cyber-Physical Systems: architecture, security and application (2019): 153-185.

[2]. Verma, Anurag, Surya Prakash, Vishal Srivastava, Anuj Kumar, and Subhas Chandra Mukhopadhyay. "Sensing, controlling, and IoT infrastructure in smart building: A review." IEEE Sensors Journal 19, no. 20 (2019): 9036-9046.

[3]. Abomhara, Mohamed, and Geir M. Køien. "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks." Journal of Cyber Security and Mobility (2015): 65-88.

[4]. Krishna, Ritika Raj, Aanchal Priyadarshini, Amitkumar V. Jha, Bhargav Appasani, Avireni Srinivasulu, and Nicu Bizon. "State-of-the-art review on IoT threats and attacks: Taxonomy, challenges and solutions." Sustainability 13, no. 16 (2021): 9463.

[5]. Malhotra, Parushi, Yashwant Singh, Pooja Anand, Deep Kumar Bangotra, Pradeep Kumar Singh, and Wei-Chiang Hong. "Internet of things: Evolution, concerns and security challenges." Sensors 21, no. 5 (2021): 1809.

[6]. Rethinavalli, S., and R. Gopinath. "Botnet attack detection in internet of things using optimization techniques." International Journal of Electrical Engineering and Technology 11, no. 10 (2020): 412-420.

[7]. Soe, Yan Naung, Yaokai Feng, Paulus Insap Santosa, Rudy Hartanto, and Kouichi Sakurai. "Machine learning-based IoT-botnet attack detection with sequential architecture." Sensors 20, no. 16 (2020): 4372.

[8]. Al-Sarem, Mohammed, Faisal Saeed, Eman H. Alkhammash, and Norah Saleh Alghamdi. "An aggregated mutual information based feature selection with machine learning methods for enhancing IoT botnet attack detection." Sensors 22, no. 1 (2021): 185.

[9]. Guerra, Alejandro. "FEATURE SELECTION FOR MACHINE LEARNING BASED IOT BOTNET ATTACK DETECTION."

[10]. Sriram, S., R. Vinayakumar, Mamoun Alazab, and K. P. Soman. "Network flow based IoT botnet attack detection using deep learning." In IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS), pp. 189-194. IEEE, 2020.

[11]. Popoola, Segun I., Bamidele Adebisi, Ruth Ande, Mohammad Hammoudeh, and Aderemi A. Atayero. "Memory-efficient deep learning for botnet attack detection in IoT networks." Electronics 10, no. 9 (2021): 1104.

[12]. Stiawan, Deris, Abdi Bimantara, Mohd Yazid Idris, and Rahmat Budiarto. "IoT botnet attack detection using deep autoencoder and artificial neural networks." KSII Transactions on Internet & Information Systems 17, no. 5 (2023).

[13]. Popoola, Segun I., Ruth Ande, Bamidele Adebisi, Guan Gui, Mohammad Hammoudeh, and Olamide Jogunola. "Federated deep learning for zero-day botnet attack detection in IoT-edge devices." IEEE Internet of Things Journal 9, no. 5 (2021): 3930-3944.

[14]. Popoola, Segun I. "Federated deep learning for botnet attack detection in IoT networks." PhD diss., Manchester Metropolitan University, 2022.

[15]. Dharini, N., S. P. Shakthi, and S. S. Shruthi. "Botnet Attack Detection in IoT-Based Security Camera Device

Using Principal Component Analysis with Various Machine Learning Algorithms." In International Conference on Information and Management Engineering, pp. 653-667. Singapore: Springer Nature Singapore, 2022.

[16]. Azhari, Revaldi Gilang, Vera Suryani, Rizka Reza Pahlevi, and Aulia Arif Wardana. "The Detection of Mirai Botnet Attack on the Internet of Things (IoT) Device Using Support Vector Machine (SVM) Model." In 2022 10th International Conference on Information and Communication Technology (ICoICT), pp. 397-401. IEEE, 2022.

[17]. Kalidindi, Archana, and Mahesh Babu Arrama. "Enhancing IoT Security with Deep Stack Encoder using Various Optimizers for Botnet Attack Prediction." International Journal of Advanced Computer Science and Applications 14, no. 6 (2023).

[18]. Sangher, Kanti Singh, Archana Singh, Hari Mohan Pandey, and Lakshmi Kalyani. "Implementation of threats detection modeling with Deep learning in IoT botnet attack environment." In IOT with Smart Systems: Proceedings of ICTIS 2022, Volume 2, pp. 585-592. Singapore: Springer Nature Singapore, 2022.

[19]. Popoola, Segun, Bamidele Adebisi, Guan Gui, Mohammad Hammoudeh, Haris Gacanin, and Darren Dancey. "Optimizing deep learning model hyperparameters for botnet attack detection in iot networks." IEEE Internet of Things Journal (2022).

[20]. Negera, Worku Gachena, Friedhelm Schwenker, Taye Girma Debelee, Henock Mulugeta Melaku, and Yehualashet Megeresa Ayano. "Review of botnet attack detection in SDN-enabled IoT Using machine learning." Sensors 22, no. 24 (2022): 9837.

[21]. Masoodi, Peerzada Saima Rafiq. "IoT Botnet Attack Detection and Comparison of Various Ensembler Learning Techniques/Models using Machine Learning." In 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), pp. 578-585. IEEE, 2021.

[22]. Jeyanthi, D. V., and B. Indrani. "Botnet Attack Detection in Iot Networking Environment Using Machine Learning Approach." AIJR Abstracts (2022): 28.

[23]. Alissa, Khalid, Tahir Alyas, Kashif Zafar, Qaiser Abbas, Nadia Tabassum, and Shadman Sakib. "Botnet attack detection in iot using machine learning." Computational Intelligence and Neuroscience 2022 (2022).

[24]. Al-Marsoomi, Omer Adil Hussein. "DDOS attack by botnet infected IoT devices detected based on deep learning models." Master's thesis, Altınbaş Üniversitesi/Lisansüstü Eğitim Enstitüsü, 2022.

[25]. Batham, Yashi, and Rakesh Kumar Tiwari. "A CNN Deep Learning Technique for Botnet Attack Detection for IoT Application." In 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN), pp. 1171-1176. IEEE, 2023.

[26]. Borra, Subba Reddy, B. Swathi Akshaya, B. Sraveena, and B. Satya Sahithi. "Machine Learning Algorithms-based Prediction of Botnet Attack for IoT devices." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 14, no. 03 (2023): 65-78.

[27]. Karaca, Kübra Nilgün, and Aydın Çetin. "Botnet attack detection using convolutional neural networks in the IoT environment." In 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1-6. IEEE, 2021.

[28]. Li, Zewen, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE transactions on neural networks and learning systems (2021).

[29]. Cheng, Jianpeng, Li Dong, and Mirella Lapata. "Long short-term memory-networks for machine reading." arXiv preprint arXiv:1601.06733 (2016).

[30]. Chen, Xue-wen, and Jong Cheol Jeong. "Enhanced recursive feature elimination." In Sixth international conference on machine learning and applications (ICMLA 2007), pp. 429-435. IEEE, 2007.

[31]. Berrar, Daniel. "Cross-Validation." (2019): 542-545.

[32]. García, María Vega, and José L. Aznarte. "Shapley additive explanations for NO2 forecasting." Ecological Informatics 56 (2020): 101039.

[33]. Rigatti, Steven J. "Random forest." Journal of Insurance Medicine 47, no. 1 (2017): 31-39.

[34]. N. Moustafa and J. Slay, “Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network dataset),” in 2015 Military Communications and Information Systems Conference (MilCIS), Nov 2015, pp. 1–6.

[35]. Fuhrer, Claus, Jan Erik Solem, and Olivier Verdier. Scientific Computing with Python: High-performance scientific computing with NumPy, SciPy, and pandas. Packt Publishing Ltd, 2021.