WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment

[1]Fawaz Ahmed Khan, [2]Mohammed Haroon Rasheed, [3]Mohammed Ali Hasan

[123]Department of IT, Nawab Shah Alam Khan College of Engineering and Technology, Hyderabad, India.

A B S T R A C T:

This research paper presents the design and implementation of a hybrid machine learning-based botnet attack detection system tailored for Internet of Things (IoT) environments. The proposed system integrates multiple deep learning architectures—Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN)—to analyze and classify network traffic for identifying botnet activity. Leveraging the UNSW-NB15 dataset, the model extracts meaningful spatial and temporal features to distinguish between normal and malicious traffic. The hybrid ACLR model is trained and evaluated using standard classification metrics such as accuracy, precision, recall, and F1-score. Feature preprocessing techniques, including normalization and label encoding, enhance data representation and improve model performance. Designed for scalability and real-time response, the system aims to reduce false positives and support secure IoT deployments. This work highlights the effectiveness of deep learning-driven hybrid architectures in fortifying IoT networks against evolving cyber threats.

Keywords: Botnet Detection, IoT Security, ACLR Model, UNSW-NB15 Dataset, CNN-LSTM-RNN-ANN Integration, Feature Preprocessing, Cyber Threat Classification

Introduction:

The rapid proliferation of Internet of Things (IoT) devices has revolutionized various sectors, including healthcare, smart homes, industrial automation, and transportation. However, the interconnected nature and limited security capabilities of these devices have also made IoT networks attractive targets for cybercriminals, particularly through botnet-based attacks. Botnets exploit vulnerable devices to perform large-scale malicious activities such as Distributed Denial-of-Service (DDoS) attacks, data breaches, and unauthorized access. Traditional intrusion detection systems often struggle to detect complex and evolving threats in IoT environments due to the sheer volume and variability of network traffic. This has led to an increasing need for intelligent, adaptive detection frameworks that can analyse network behaviour and accurately identify botnet activity. In response, this project introduces a hybrid deep learning-based detection model that leverages the strengths of multiple neural architectures to effectively classify and mitigate botnet threats in real time.

Literature Review:

The surge in IoT device usage has prompted extensive research into safeguarding these networks from emerging threats, especially botnet attacks. Researchers have focused on utilizing machine learning and deep learning techniques to develop intelligent intrusion detection systems capable of handling complex, high-dimensional network traffic. The following studies provide foundational insights and advancements relevant to the proposed hybrid model:

- ***M. M. Rathore, A. Paul, A. Ahmad, S. Rho (2018)***
  Proposed a deep learning-based botnet detection framework for IoT using communication graph construction and stacked autoencoders (SAE). The model demonstrated high accuracy in identifying both known and zero-day attacks with minimal overhead, making it suitable for resource-constrained environments.
- ***Meidan, Yair, et al. (2018)***
  Developed N-BaIoT, a dataset for IoT botnet detection, and employed deep autoencoders to detect anomalies in network traffic generated by infected devices. The model achieved over 99% accuracy, showcasing the effectiveness of deep learning in real-world scenarios.
- ***A. Saied, R. E. Overill, T. Radzik (2016)***
  Presented a botnet detection system using supervised machine learning models with feature extraction based on flow-based traffic characteristics. It demonstrated the importance of selecting meaningful traffic features to improve classification accuracy.
- **Yin, Chuanlong, et al. (2017)**
  Introduced a deep learning model combining CNN and RNN to detect cyberattacks in network traffic data. Their hybrid approach effectively captured spatial and temporal patterns, achieving superior detection rates compared to traditional methods.

Methodology:

The botnet detection system was developed using Python and deep learning frameworks, employing a hybrid architecture that integrates multiple neural networks. The methodology involved the following key steps:

3.1 Setting Up the Environment

The development environment was configured with essential Python libraries for data handling, preprocessing, and model building. Key libraries included NumPy, Pandas, Scikit-learn, TensorFlow, and Keras. The system was designed to process network traffic data in CSV format, sourced from the UNSW-NB15 dataset.

3.2 Dataset Collection and Preparation

The UNSW-NB15 dataset, which contains labeled network traffic instances representing both benign and various attack types (including botnets), was used for training and evaluation. The dataset was structured to include multiple features related to traffic behavior, protocol usage, and connection patterns.

3.3 Feature Extraction and Preprocessing

Features relevant to identifying botnet activity were selected and preprocessed. This included label encoding of categorical attributes and normalization of numerical values to ensure uniform input to the models. The dataset was then split into training and testing subsets to evaluate model generalizability.

3.4 Model Architecture and Integration (ACLR)

A hybrid architecture—referred to as ACLR—was constructed by integrating four deep learning models: Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory networks (LSTM). Each model was trained on the preprocessed data to learn specific spatial or temporal features.

3.5 Model Training and Evaluation

All models were trained using the training set and evaluated on the testing set. Key performance metrics such as accuracy, precision, recall, and F1-score were used to assess the effectiveness of each individual model and the final hybrid model. The ACLR model showed superior detection capability with reduced false positives.

3.6 Botnet Attack Prediction

The trained ACLR model was deployed for real-time prediction. Incoming network traffic data, once preprocessed, was passed through the model to determine whether it represented normal behavior or a botnet attack. The system is optimized for fast response, enabling integration with live network monitoring tools for proactive threat mitigation.

4. Illustrations:

```
68  @app.route("/data_preprocessing",methods =["GET", "POST"] )
69  def data_preprocessing():
70      fname = request.form.get("file")
71      x_train, x_test, y_train, y_test = training_features(fname)
72      dict['x_train'] = x_train
73      dict['y_train'] = y_train
74      dict['x_test'] = x_test
75      dict['y_test'] = y_test
76      return render_template("data_preprocessing.html",msg="success")
77
```

Fig. 1 – Data Preprocessing

```
@app.route("/evaluations")
def evaluations():

    hydrid_list = []
    lstm_list = []
    cnn_list = []
    rnn_list = []
    ann_list =[]
    metrics = []

    x_train = dict['x_train']
    y_train = dict['y_train']
    x_test = dict['x_test']
    y_test = dict['y_test']
    #HYBRID
    accuracy_hybrid,precision_hybrid,recall_hybrid,fscore_hybrid = hybrid_evaluation(x_train, x_test, y_train, y_test)
    hydrid_list.append("ACLR")
    hydrid_list.append(accuracy_hybrid)
    hydrid_list.append(precision_hybrid)
    hydrid_list.append(recall_hybrid)
    hydrid_list.append(fscore_hybrid)
    # lstm
    accuracy_lstm, precision_lstm, recall_lstm, fscore_lstm = lstm_evaluation(x_train, x_test, y_train, y_test)
    lstm_list.append("LSTM")
    lstm_list.append(accuracy_lstm)
    lstm_list.append(precision_lstm)
    lstm_list.append(recall_lstm)
    lstm_list.append(fscore_lstm)
    # cnn
    accuracy_cnn,precision_cnn,recall_cnn,fscore_cnn = cnn_evaluations(x_train, x_test, y_train, y_test)
    cnn_list.append("CNN")
    cnn_list.append(accuracy_cnn)
    cnn_list.append(precision_cnn)
    cnn_list.append(recall_cnn)
    cnn_list.append(fscore_cnn)
    #rnn
    accuracy_rnn,precision_rnn,recall_rnn,fscore_rnn = rnn_evaluations(x_train, x_test, y_train, y_test)
    rnn_list.append("RNN")
    rnn_list.append(accuracy_rnn)
    rnn_list.append(precision_rnn)
    rnn_list.append(recall_rnn)
    rnn_list.append(fscore_rnn)
    #
```

Fig. 2.1 – Model Evaluation

```
    accuracy_ann, precision_ann, recall_ann, fscore_ann = ann_evaluations(x_train, x_test, y_train, y_test)
    ann_list.append("ANN")
    ann_list.append(accuracy_ann)
    ann_list.append(precision_ann)
    ann_list.append(recall_ann)
    ann_list.append(fscore_ann)

    metrics.clear()
    metrics.append(hydrid_list)
    metrics.append(lstm_list)
    metrics.append(cnn_list)
    metrics.append(rnn_list)
    metrics.append(ann_list)
    return render_template("evaluations.html", evaluations=metrics)
```

Fig. 2.2– Model Evaluation

```
def recall_graph():
    db = DBConnection.getConnection()
    cursor = db.cursor()
    recall_list.clear()
    cursor.execute("select recall from evaluations")
    recdata = cursor.fetchall()
    for record in recdata:
        recall_list.append(float(record[0]))
    height = recall_list
    bars = ('ACLR','LSTM', 'CNN', 'RNN','ANN')
    y_pos = np.arange(len(bars))
    plt3.bar(y_pos, height, color=['black','green', 'blue', 'red','pink'])
    plt3.xticks(y_pos, bars)
    plt3.xlabel('Algorithms')
    plt3.ylabel('Recall')
    plt3.title('Analysis on DL Recall')
    plt3.savefig('static/recall.png')
    plt3.clf()
    return ""

def f1score_graph():
    db = DBConnection.getConnection()
    cursor = db.cursor()
    f1score_list.clear()
    cursor.execute("select f1score from evaluations")
    fsdata = cursor.fetchall()
    for record in fsdata:
        f1score_list.append(float(record[0]))
    height = f1score_list
    bars = ('ACLR','LSTM', 'CNN', 'RNN','ANN')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height, color=['red', 'green', 'blue','cyan','brown'])
    plt.xticks(y_pos, bars)
    plt.xlabel('Algorithms')
    plt.ylabel('F1-Score')
    plt.title('Analysis on DL F1-Score')
    plt4.savefig('static/f1score.png')
    plt4.clf()
    return ""
#user section
```

Fig. 3 – Performance Evaluation

5. Result:

The proposed system effectively detects botnet attacks in IoT environments by analysing network traffic features using a hybrid deep learning model. The integration of ANN, CNN, RNN, and LSTM architectures (ACLR model) enables accurate classification of complex attack patterns without requiring manual rule-based configurations. When evaluated on the UNSW-NB15 dataset, the model demonstrated high detection accuracy with strong precision, recall, and F1-score values. Among the individual models, the CNN-LSTM combination exhibited notable performance in capturing both spatial and temporal patterns. The hybrid ACLR model further enhanced overall accuracy and reduced false positives, making it suitable for real-time deployment. These results validate the system's effectiveness in improving the security posture of IoT networks and its potential for practical application in live intrusion detection scenarios.

6. Requirements:

6.1. Hardware Requirements

Processor	:	Any Update Processer
Ram	:	Min 4 GB
Hard Disk	:	Min 250 GB

6.2. Software Requirements

Operating System	:	Windows family
Technology	:	Python 3.8
Front-end Technology	:	HTML, CSS, JS
Back-end Technology	:	MySQL
IDE	:	PyCharm IDE
Web Framework	:	Flask

7. Conclusion:

This paper presents a comprehensive approach to detecting botnet attacks in IoT environments using a hybrid deep learning architecture. By integrating ANN, CNN, RNN, and LSTM models, the system effectively analyses network traffic patterns to identify malicious activities in real time. The project demonstrates the practical viability of applying advanced machine learning techniques to enhance IoT security. Through extensive evaluation using the UNSW-NB15 dataset, the proposed model achieves high accuracy and reliability while maintaining scalability for deployment in real-world networks. This work underscores the critical role of intelligent, automated detection systems in protecting IoT infrastructure from evolving cyber threats.

Appendix A. Detailed Algorithm

Step 1: Import Required Libraries

- Import essential Python libraries such as NumPy and Pandas for data manipulation.
- Use Scikit-learn for preprocessing and evaluation, and TensorFlow/Keras for deep learning model development.

Step 2: Dataset Collection and Organization

- Use the publicly available UNSW-NB15 dataset, which includes both benign and attack traffic data.
- Organize the dataset into labelled CSV files, with relevant traffic features and their corresponding class labels.

Step 3: Feature Extraction and Engineering

- Load and inspect the dataset.
- Extract important features from the traffic data, including flow-based and packet-level characteristics.
- Apply feature encoding (e.g., label encoding for categorical features) and normalization for uniform input across models.

Step 4: Data Preprocessing

- Clean the dataset by handling missing values and removing redundant features.
- Normalize or standardize the feature values.
- Split the dataset into training and testing sets (commonly 70% training, 30% testing).

Step 5: Model Training (ACLR Architecture)

- Train four deep learning models:
- Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)
- Integrate these into a hybrid architecture (ACLR) to leverage their strengths.
- Tune hyperparameters and use techniques like dropout and batch normalization to prevent overfitting.

Step 6: Evaluation

- Evaluate each model and the combined ACLR model using test data.
- Use performance metrics such as:
- Accuracy
- Precision
- Recall
- F1-Score
- Optionally, visualize performance using confusion matrices or ROC curves.

Step 7: Prediction Functionality

- Create a prediction pipeline to accept new traffic input in CSV format
- Preprocess the input data to match the trained model's format.
- Feed the processed input to the trained ACLR model for real-time botnet detection.
- Output whether the traffic is benign or botnet-infected.
- Step 8: Model Saving and Reuse
- Save trained models using model.save() (Keras) or joblib for reuse.
- Load models in future sessions without retraining using load_model() or joblib's load function.

Appendix B. Survey Questionnaire

The following questionnaire was used to gather feedback on the botnet detection system:

- How easy was it to upload and process network traffic data using the system?
- Was the interface for submitting traffic data and retrieving predictions user-friendly?
- Were the prediction results (botnet or benign) clearly presented and understandable?
- Did the system provide helpful feedback in case of incorrect input format or errors?
- Do you feel confident in the system's ability to accurately detect botnet threats in IoT traffic?

REFERENCES:

1. M. M. Rathore, A. Paul, A. Ahmad, and S. Rho, "Distributed denial of service attack detection system using hybrid machine learning algorithms," *Computers & Electrical Engineering*, vol. 70, pp. 354–368, Aug. 2018.
2. Y. Meidan, M. Bohadana, A. Shabtai, J. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
3. A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," *Neurocomputing*, vol. 172, pp. 385–393, Jan. 2016.
4. C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
5. T. S. Gunther and B. E. Mullins, "Anomaly detection in IoT traffic using recurrent neural networks," in *Proc. MILCOM 2018 - IEEE Military Communications Conference*, Los Angeles, CA, USA, 2018, pp. 1–6.
6. F. Javed, M. K. Afzal, M. Sharif, and B. Kim, "Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2062–2100, 2018.
7. K. Doshi, Y. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2018, pp. 29–35.
8. F. A. Ghaleb, K. M. Faraoun, and A. Boukhtouta, "Hybrid intrusion detection system using supervised learning techniques," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 2, pp. 91–103, 2019.
9. G. Hoang, T. T. Nguyen, and D. T. Tran, "An efficient combination of CNN and LSTM for botnet detection using network traffic," *International Journal of Network Security*, vol. 23, no. 1, pp. 94–102, Jan. 2021.
10. J. Zhang, M. Zulkernine, and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, Sept. 2008.