Meta-Learning-based Traffic Classification for IoT Devices

Saeid Rajabi
srajabi@udel.edu
University of Delaware
Newark, Delaware, USA

Abstract

This project developed a machine learning-based system to classify IoT network traffic into benign and attack types. Using a diverse dataset from multiple IoT devices, we implemented Logistic Regression, Decision Trees, Random Forests, and a Multi-Layer Perceptron (MLP) meta-learner. While binary classification achieved near-perfect accuracy, multiclass classification highlighted challenges with class feature overlap, particularly for class 4. By addressing these issues with preprocessing and class balancing, we improved overall performance. This work demonstrates the effectiveness of machine learning in IoT security and sets the stage for future advancements in real-time and scalable intrusion detection systems.

1. Introduction

The explosion of Internet of Things (IoT) devices, such as smart thermostats, fitness trackers, and connected appliances, has transformed how we live, making our homes and workplaces more connected and automated. However, this rapid growth also presents serious security risks. Many of these devices have limited processing power and often use outdated or weak security measures, making them easy targets for hackers.

When malicious actors target IoT devices, they can disrupt their normal functions, interfere with essential services, or even hijack them to launch massive cyberattacks known as Distributed denial-of-service (DDoS) attacks. These attacks can overwhelm networks, causing widespread outages and chaos.

One of the biggest challenges in protecting IoT networks is telling the difference between regular, harmless traffic and dangerous, malicious activity. This is especially tough because IoT data can be complex and unbalanced; there's usually more regular traffic than attack attempts. Different types of cyberattacks can look very similar, making it hard to identify and classify them accurately.

To tackle these issues, cybersecurity experts are working on advanced machine-learning models. These intelligent systems are designed to sift through the vast amounts of IoT data, spot potential threats, and keep our connected devices and networks safe and reliable. Ensuring IoT systems' security and smooth operation is crucial as we continue to integrate these smart technologies into every aspect of our lives.

Contribution. This project aims to address the challenges of IoT security through an AI-based framework that leverages MLPs for both binary and multiclass classification of network traffic. By integrating base models and meta-learning, the framework seeks to enhance detection accuracy, particularly for underrepresented and challenging attack classes. The main **contributions** are as follows.

- Developing a robust dataset by preprocessing and balancing IoT traffic data to address class imbalance.
- Evaluating base models (e.g., Logistic Regression, Decision Tree, Random Forest) to identify strengths and weaknesses.
- Designing an MLP-based meta-learner to integrate base model predictions, improving classification performance.

Significance of the study. This work contributes to the broader field of IoT security by tackling key issues in traffic classification, including feature overlap, data imbalance, and scalability. The findings demonstrate the potential of meta-learning in combining the strengths of diverse models to achieve higher accuracy and robustness. Furthermore, this project underscores the importance of balancing automation with interpretability, ensuring that AI-driven security solutions remain actionable and aligned with real-world requirements.

2. Background

Numerous factors undermine the security of Internet of Things (IoT) systems. Traditional network security best practices often encounter significant challenges when applied to IoT networks. These challenges include:

1. **Limited Power.** Numerous critical IoT devices, including sensors and actuators, function with low power. This limitation affects the sophistication of the algorithms they can deploy, constraining their ability to support advanced security protocols.

2. **Insufficient Storage.** IoT devices often have insufficient storage capacity, limiting their ability to retain and process data, which hinders the implementation of robust security protocols.

3. **Human Vulnerabilities.** A frequent security vulnerability occurs when users fail to change the factory default or hard-coded usernames and passwords on their IoT devices. This oversight creates a simple access route for malicious actors.

Recently, two well-known botnets exploited these security weaknesses in IoT devices to execute distributed denial of service (DDoS) attacks. These botnets, *Mirai* and *Bashlite* (also known as *Gafgyt*), took advantage of vulnerable devices to orchestrate large-scale cyberattacks.

Mirai functions by continuously searching the Internet for IoT devices that rely on default or hard-coded usernames and passwords. Upon discovering such a device, *Mirai* infects it with malware that instructs the device to connect with a central control server, effectively transforming it into a bot. These compromised bots can then be utilized to execute powerful DDoS attacks. Generally, *Mirai* focuses on routers, DVRs, CCTV cameras, and other smart, internet-connected devices, taking advantage of their weaknesses to form a significant botnet.

The use of IoT devices by botnets such as *Mirai* underscores the critical necessity for improved security protocols specifically designed for IoT networks. It is vital to tackle the technical shortcomings of devices while also enhancing user practices to safeguard these interconnected systems against upcoming cyber threats.

3. Methodology

This section outlines our approach to managing the classification task within our dataset. We begin by exploring the dataset's structure, features, and the preprocessing techniques used to ready the data for analysis. Following this, we discuss the design and architecture of the machine learning framework, highlighting the base models and the meta-learner applied to enhance classification performance. Finally, we detail the implementation specifics, including the training process and the optimizations implemented to increase the model's accuracy and robustness.

3.1. Dataset

The dataset used for this project captures network traffic data from **nine IoT devices** infected with the **Mirai** and **Bashlite (Gafgyt)** botnets [1].

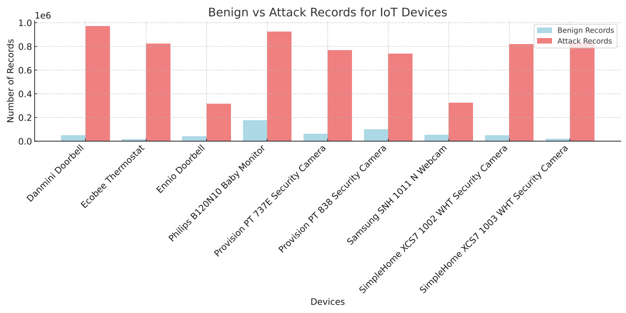


Figure 1. Number of records in different types of attacks

The devices include commonly found IoT hardware, such as doorbells, security cameras, baby monitors, thermostats, and webcams. The dataset contains traffic from **ten distinct attack types** alongside benign traffic, forming an **11-class classification problem** Fig. 1.

The attack traffic is categorized as follows:

- **Mirai Attacks:**
  - **Scan:** Automated scanning of vulnerable devices.
  - **Ack:** ACK flooding aimed at overwhelming the network.
  - **Syn:** SYN flooding to exploit TCP handshake vulnerabilities.
  - **UDP:** Standard UDP flooding attacks.
  - **UDPplain:** Optimized UDP flooding for higher packets-per-second (PPS).
- **Bashlite Attacks:**
  - **Scan:** Network scanning for vulnerable devices.
  - **Junk:** Sending spam data to targets.
  - **UDP:** UDP flooding targeting specific devices.
  - **TCP:** TCP flooding attacks.
  - **COMBO:** Combined spam and targeted connection flooding.

The dataset provides two notable advantages:

- **Network-based Capture:** The traffic was recorded at the network level, offering a broader perspective compared to host-based datasets.
- **Real Devices:** Data was collected from real IoT devices, ensuring realistic attack scenarios.

The dataset includes **115 statistical features**, such as mean, variance, magnitude, frequency, and covariance.

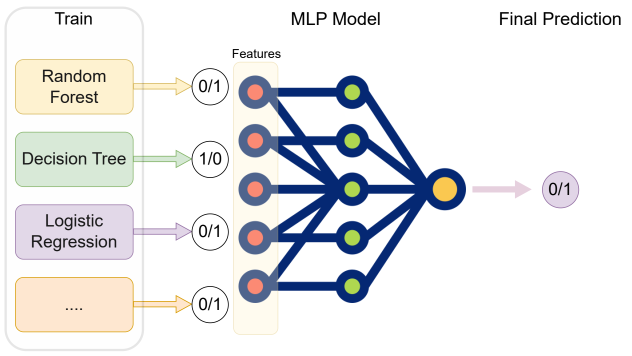


Figure 2. Proposed meta-learner architecture

3.2. Model Design

The machine learning framework employs a two-layer approach with **base models** and a **meta-learner**:

3.2.1 Base Models

Three base models were selected to generate prediction probabilities for each class:

- Logistic Regression: Chosen for its simplicity and efficiency in modeling linear relationships.
- Decision Trees: Used for their interpretability and ability to model non-linear data.
- Random Forests: Leveraged as an ensemble of Decision Trees for robustness and generalization.

3.2.2 Meta-Learner

As shown in Fig. 2, the **Multi-Layer Perceptron (MLP)** meta-learner aggregates predictions from the base models. Its architecture comprises:

- **Three Hidden Layers:** Neuron sizes of 32-16-8 designed to balance model complexity and computational efficiency.
- **Softmax Output Layer:** Converts logits into probabilities for multiclass classification.
- **Regularization:** Techniques such as dropout (rate = 0.2) and L2 regularization to prevent overfitting.
- **Loss Function:** Cross-Entropy Loss to optimize classification performance.

3.3. Implementation & Training

The implementation was carried out using Python and libraries like **scikit-learn** [2] and **PyTorch** [3].

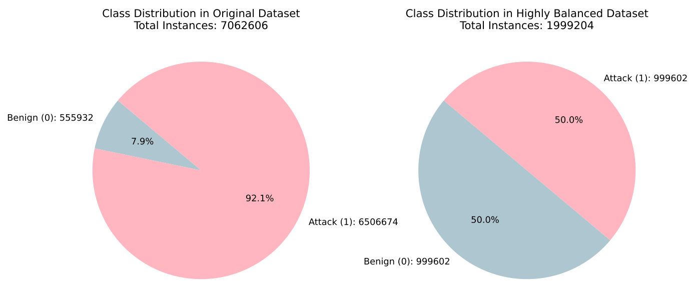


Figure 3. Class distributions before VS. after balacing

3.3.1 Preprocessing

- **Standardization:** Features were standardized using z-score normalization.
- **Data Balancing:** Synthetic minority oversampling technique (SMOTE) was applied to augment under-represented classes. As shown in 3, the classes were highly imbalanced by having around 8% of benign traffic and more than 92% of attack traffic, but after balancing, this ratio became 50% for each classes.
- **Data Splitting:** Using K-Fold Cross-Validation with three folds ensures balanced distribution in the training and validation sets.
- **Dataset Statistics:**

Table 1. Dataset statistics on some features

Feature	**Mean**	**Median**	**Mode**	**Range**
MI_dir_L5_weight	81.65	68.94	3138.76	261.29
MI_dir_L5_mean	178.13	167.31	9428.80	681.64
MI_dir_L5_variance	15383.57	15097.06	1052287.81	154122.32
MI_dir_L3_weight	129.20	109.56	5220.99	365.02
MI_dir_L3_mean	178.23	172.09	10301.65	603.21
MI_dir_L3_variance	17061.87	17285.07	858729.59	150873.96
MI_dir_L1_weight	367.05	316.11	14593.59	812.52
MI_dir_L1_mean	178.48	179.48	9745.88	480.71
MI_dir_L1_variance	18502.39	19367.82	522951.54	119373.83
MI_dir_L0.1_weight	3397.75	3026.49	154210.40	5772.74
MI_dir_L0.1_mean	178.85	185.73	8919.95	331.49
MI_dir_L0.1_variance	19209.55	20983.96	645031.62	92588.98
MI_dir_L0.01_weight	19722.85	18878.05	1090856.62	34552.47
MI_dir_L0.01_mean	178.74	187.50	9295.76	289.57
MI_dir_L0.01_variance	19456.00	21366.15	877342.62	75436.52
H_L5_weight	81.65	68.94	3138.76	261.29
H_L5_mean	178.13	167.31	9428.80	681.64
H_L5_variance	15383.57	15097.06	1052287.81	154122.32

- **Encoding:** For non-numerical values, we could get the

3.3.2 Base Model Training

The base models were trained on standardized features. Their prediction probabilities served as input meta-features for the MLP meta-learner. Hyperparameters were tuned to optimize performance:

- Logistic Regression: Increased `max_iter` to 200 for convergence.

- Random Forests: Limited tree depth to reduce computational cost.
- Decision Tree: Decrease the max depth to 10.

3.3.3 Meta-Learner Training

- **Optimizer:** Adam optimizer with a learning rate of 0.001.
- **Training Epochs:** The MLP was trained for 50 epochs, balancing convergence and overfitting.

This methodology effectively combined the strengths of base models while addressing their limitations through the meta-learner, achieving improved performance on challenging classes.

4. Results

4.1. Evaluation Metrics

Accuracy, precision, recall, and F1-score were used for evaluation, complemented by confusion matrices for class-specific analysis.

4.2. Experiments

In this project, we performed two types of experiments, which are binary class classification and multi-class classification.

Since the dataset was linearly separable, the base models could perform well. We wanted to show the power of meta-learner to see if it can learn from poor performance base models. To do so, we introduced controllable noise in around 30% to 40%.

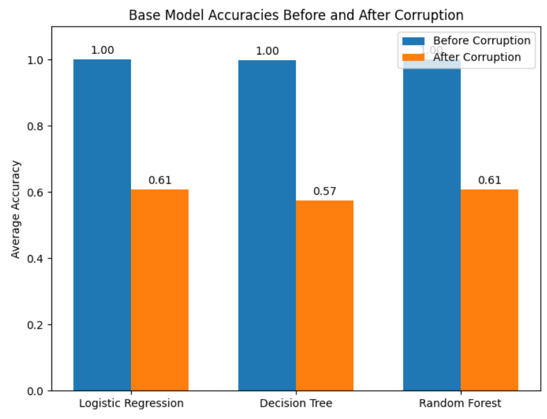


Figure 4. Base-models performance before and after corruption.

Figure 4, shows the accuracy of the base models before and after adding noise.

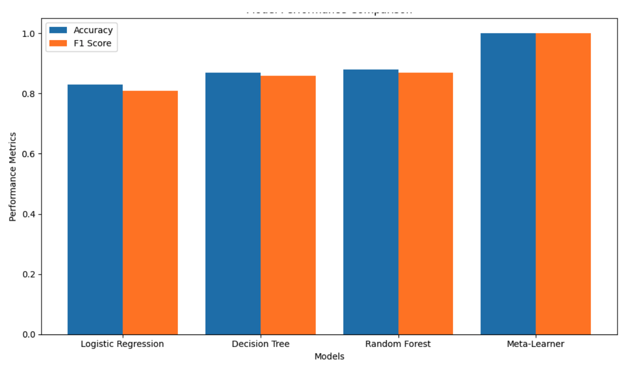


Figure 5. Comparison of base models and meta-learner

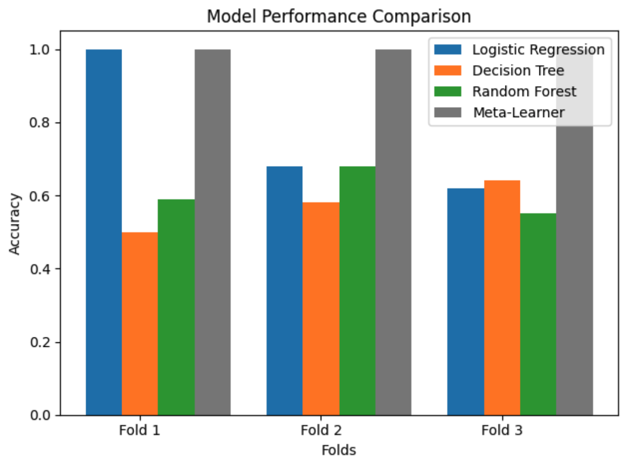
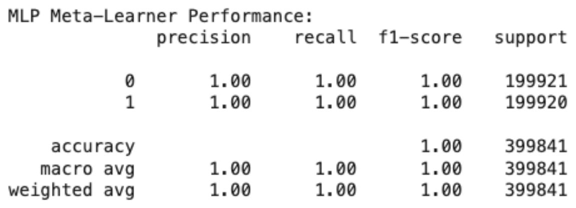


Figure 6. Comparison of each models in each folds



```
MLP Meta-Learner Performance:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    199921
           1       1.00      1.00      1.00    199920

    accuracy                           1.00    399841
   macro avg       1.00      1.00      1.00    399841
weighted avg       1.00      1.00      1.00    399841
```

Figure 7. Meta-learner training report

As shown in Fig. 5, we can observe that, meta-learner could learn even from the corrupted base models and could achieve the 100% accuracy and F1-score compared to other base models.

Because the dataset in this experiment was quite large, we needed to use the K-fold approach, and in our case, K is 3.

Figure 6 shows the performance of each model in each fold and the meta-learner outperforms the other model performances in all the folds, while Fig. 7 shows the report for meta-learner with 100% accuracy.

Figure 9 shows the decrease of the loss over the epochs.

In the multi-class classification task, class 4 consistently

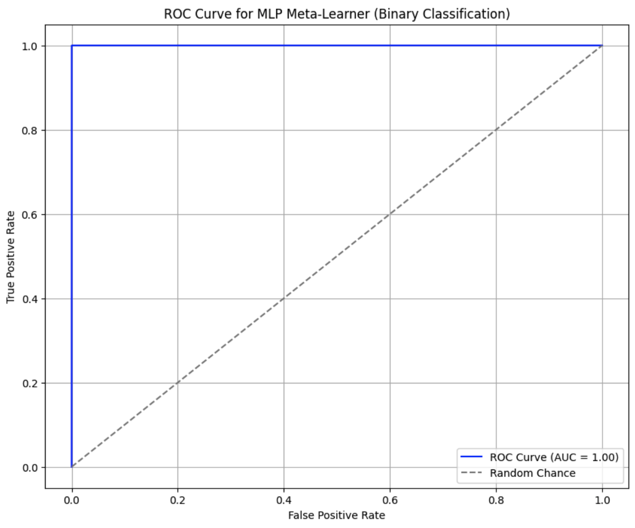


Figure 8. ROC curve for binary classification

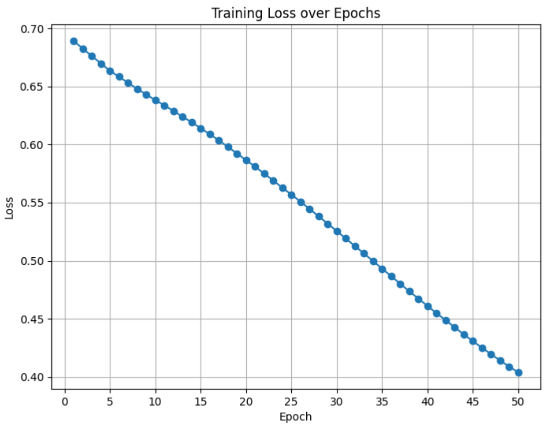


Figure 9. Training loss over epochs

posed significant challenges. These issues were particularly pronounced when we analyzed its recall, which consistently remained at zero across all models. This indicates that the models struggled to correctly identify instances belonging to this class.

A deeper investigation into the dataset and feature distribution revealed that class 4 shared overlapping feature spaces with other classes. This overlap likely caused confusion during classification, leading to frequent misclassifications of class 4 instances as these neighboring classes. Moreover, class 4 appeared to have insufficient distinguishing features or representation in the dataset, further compounding the difficulty for the models.

These challenges emphasize the need for targeted preprocessing, such as enhanced feature engineering or oversampling, to better capture class 4's unique characteristics and improve model performance in future iterations.

Table 2. Classification Report for MLP Meta-Learner for multiclass task

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	81,464
1	1.00	1.00	1.00	76,784
2	1.00	1.00	1.00	54,375
3	1.00	1.00	1.00	75,323
4	0.74	0.00	0.00	110,874
5	0.51	1.00	0.68	115,693
6	1.00	1.00	1.00	99,279
7	1.00	1.00	1.00	76,113
8	1.00	1.00	1.00	78,619
9	1.00	1.00	1.00	92,477
10	1.00	1.00	1.00	86,241
Accuracy	0.88 (on 947,242 samples)			
Macro Avg	0.93	0.91	0.88	
Weighted Avg	0.91	0.88	0.84	

4.2.1 Final Results

- **Binary Classification:** The models achieved exceptional performance, with over 100% accuracy, precision, recall, and F1-score. Misclassifications were minimal, demonstrating strong performance in distinguishing benign from attack traffic.

- **Multiclass Classification:** Overall accuracy ranged from 83% (Logistic Regression) to 88% (Decision Tree and Random Forest). Despite strong performance across most classes, class 4 consistently got zero recall, indicating the models' inability to correctly classify any samples belonging to this class.

5. Discussion

5.1. Interpretation

Our experiments reveal both the strengths and weaknesses of the proposed methodology. Although the models excelled in binary classification with near-perfect accuracy, their performance in multiclass classification highlighted important areas for enhancement. The difficulty in accurately classifying class 4 emphasizes challenges linked to data representation and feature overlap within the dataset. These findings carry substantial real-world implications for IoT security, where dependable identification of specific attack classes is essential for maintaining system resilience. The ongoing challenges with class 4 indicate a need for improved, non-overlapping data.

5.2. Challenges & Learnings

Throughout the project, we encountered several challenges that required careful analysis and iterative refinement:

- **Class Imbalance:** Techniques like SMOTE and targeted oversampling were employed to mitigate this issue.

- **Learning Rate for Optimizer:** Adjusting the Adam optimizer's learning rate was crucial for achieving an optimal balance between convergence speed and stability. Modifications to weight decay and dropout were made as well to mitigate overfitting.

- **Encoding and Normalizing:** Encoding categorical features and normalizing numerical ones were crucial preprocessing steps that significantly impacted the models' performance.

- **Number of Epochs:** Initially, training for 50 epochs was insufficient for the models to converge effectively. Extending training to 200 epochs improved performance but increased computational demand.

- **Converting Base Model Predictions:** Integrating base model predictions as input to the MLP meta-learner was a non-trivial task that required careful design to ensure compatibility and effective information transfer.

- **Feature Overlap Between Classes:** Certain classes, such as class 4 and class 5, exhibited significant feature overlap, leading to frequent misclassifications. This challenge highlighted the need for better feature engineering and selection.

- **Large Feature Space:** The high-dimensional dataset posed challenges for both computation and model generalization, necessitating dimensionality reduction techniques like PCA.

- **Computational Limitations During Training:** Resource constraints during training required optimization of model architecture and hyperparameters to achieve the desired performance within feasible computational limits.

5.3. Insights

Several key insights emerged from this project, showing our understanding of the problem domain, dataset, and model behavior:

- **Need of high-quality data:** The data that we are working on are based on simulations. This highlights the need of high-quality data.

- **Role of Feature Representation:** Feature overlap between classes underscores the need for advanced feature extraction methods, such as autoencoders, to capture distinct characteristics for each class.

- **Integration of Base Models and Meta-Learners:** The use of base models to generate meta-features for the MLP meta-learner proved effective in improving overall performance, despite persistent challenges with specific classes.

- **Significance of Hyperparameter Tuning:** Adjusting hyperparameters such as learning rate, weight decay, and number of epochs had a substantial impact on model convergence and generalization.

- **Real-World Relevance:** The insights gained from analyzing IoT attack traffic have direct implications for developing scalable and reliable intrusion detection systems in smart environments.

6. Conclusion

This project successfully addressed the complex challenge of categorizing network traffic into various classes, distinguishing between benign and attack behaviors in IoT devices. The binary classification models exhibited outstanding accuracy, precision, recall, and F1-scores, showcasing their capability to differentiate between benign and attack traffic. Although the models performed well in multiclass classification, ongoing difficulties with class 4 classification revealed limitations in the dataset and feature representation. By utilizing a meta-learner approach, the project enhanced overall performance by taking advantage of the strengths of base models, highlighting the promise of ensemble learning techniques. The project confronted several issues, including class imbalance, feature overlap, and computational constraints, yielding important insights for designing robust machine learning systems applicable to real-world scenarios.

The project contributes to the field by advancing methods for IoT security, particularly in detecting diverse attack types in network traffic. The methodologies and findings are directly applicable to developing scalable intrusion detection systems, enhancing the resilience of IoT ecosystems against cyber threats.

7. Future Work

While this project has laid a strong foundation, there are several areas for improvement and exploration in future work:

- **Enhanced Feature Engineering:** Explore advanced feature extraction techniques such as autoencoders or domain-specific feature selection methods to improve class separability, particularly for challenging classes like class 4.

- **Advanced Model Architectures:** Investigate state-of-the-art deep learning models, such as transformers or graph neural networks, to capture complex relationships in the data.

- **Adversarial Training:** Implement adversarial training techniques to enhance the model's robustness against noisy or adversarial inputs, making it more reliable in real-world scenarios.
- **Scalability and Efficiency:** Optimize the training process for scalability, such as leveraging distributed computing frameworks or exploring quantization techniques for faster inference on edge devices.
- **Real-Time Application:** Deploy the trained models in a real-time network monitoring system to evaluate their practical performance and further refine them based on live traffic data and try to fine-tune it real-time.

References

[1] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, 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.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.