

Cross-Dataset Testing of Machine Learning Botnet Detection in IoT Networks

Full name: ALMOUSTAPHA MOAMAR AL ANSARY

Student ID: 262000643

Course: AI800 Research Methodology in Computer Science

Lecturer: Dr. Mohammed Anbar

Date: 22 August 2026

Abstract

IoT botnets are a growing cybersecurity threat, and detection research has moved from signature-based methods toward machine learning and deep learning. However, many existing models are tested only inside one dataset, so their performance across different networks is still unproven. This is a problem because real networks may have limited labelled attack data, and a model that works well in one dataset may not work as well in another. This study proposes to test that issue directly using Bot-IoT and N-BaIoT. First, both datasets will be cleaned, balanced, and aligned using shared traffic features. Then, three detection approaches will be trained: Random Forest, a deep learning model, and a transfer-learning variant using a small amount of target data. The models will be tested in two settings: same-dataset testing and cross-dataset testing. The study will measure accuracy, precision, recall, F1-score, false alarm rate, and the performance drop between the two settings. The results will also be compared with a single-dataset baseline. The expected outcome is a clear measurement of how much performance drops between Bot-IoT and N-BaIoT. Shared features and transfer learning are expected to reduce the drop, but even a large drop would be useful because it would show that single-dataset results can overstate real-world performance.

Chapter 1 : Introduction

1.1 Introduction

IoT devices are now part of normal life, from cameras, doorbells, routers and smart speakers in homes to sensors, controllers and monitoring devices in companies. Because many of these devices are cheap, always connected and not updated often, they become useful targets for attackers. When attackers take control of many devices at the same time, those devices can become a botnet. Real IoT botnet families such as Mirai and BASHLITE show how ordinary devices can be turned into tools for wider attacks, including denial-of-service attacks and other network abuse [4], [7]. Older signature-based detection is useful for attacks that are already known, but it can miss new or changed attacks because the pattern is not yet in the database. For this reason, researchers have moved toward machine learning and deep learning methods that learn patterns from network traffic instead of only matching fixed rules [1], [2], [3]. These methods can study traffic behaviour and decide whether it looks normal or suspicious. Recent studies continue to report very high accuracy. For example, newer work on tree-based and hybrid deep learning models reports strong results for IoT botnet detection [8], [9]. Some research also goes beyond accuracy and checks whether the method can run in a realistic setting. Kumar et al. [10], for example, tested an edge-gateway design and reported Raspberry Pi memory and delay costs. This makes IoT botnet detection important now because the devices are growing, the attacks are active, and the detection methods must be practical.

1.2 Background

An IoT botnet is a group of connected devices that have been infected and controlled by an attacker. These devices can include cameras, routers, sensors, doorbells and other smart devices. A botnet usually grows when infected devices scan the internet or local networks for other weak devices. For example, some attacks look for devices with default usernames and passwords, which helped botnets such as Mirai and BASHLITE spread [4], [5]. Once the attacker controls many devices, the botnet can be used for larger attacks. One common use is a distributed denial-of-service attack, where many infected devices send traffic at the same target until the service slows down or stops.

There are two main styles of detection. Signature-based detection compares traffic with known attack patterns. It can work well for known attacks, but it is weaker when the attack is new or changed. Anomaly-based or machine-learning detection works differently. It learns what normal or attack traffic looks like from data, then flags traffic that does not fit the learned pattern.

This proposal focuses on two common datasets. Bot-IoT is used in several earlier studies on IoT botnet detection [1], [2], [3]. N-BalIoT is another standard dataset, built from real traffic from nine IoT devices infected with Mirai and BASHLITE, and it is used by several studies in this review [4], [7], [8], [9].

Researchers usually report accuracy, precision, recall, F1-score and false positive rate. Accuracy alone is not enough because a model can look strong overall but still miss rare attacks or create too many false alarms.

1.3 Problem Statement

This field is cybersecurity, mainly botnet detection in IoT networks using machine learning and deep learning. IoT means connected devices like cameras, sensors, and smart devices.

The problem today is that many models are tested on only one dataset. So the model may learn that dataset very well, but we do not know if it can handle a different network. High accuracy in one test does not always mean it will work in real life. In real networks, labelled attack data is limited, so retraining a model for every new network can be difficult.

This can hurt people, companies, and service providers who depend on connected devices. The number of IoT devices keeps growing, so attackers have more devices and connections to target. If attacks are missed, devices can be taken over, data can be stolen, and services can stop working. If there are too many false alarms, security teams waste time and may lose trust in the system.

Paper [1] used deep learning on the Bot-IoT dataset and used SMOTE to deal with rare attacks, but it did not prove the model works on a different dataset. Paper [2] used a CNN model called BotIDS and tested it only on the Bot-IoT dataset, so its results may not transfer to another network. Paper [3] also tested a deep learning model on Bot-IoT, but it still stayed inside one dataset. Paper [4] used simpler tree-based models on the N-BalIoT dataset and still got strong

results, but it only covered two botnet families. Paper [5] focused on early detection before the full attack runs, using sandbox behaviour for about three minutes per sample, but it tested outside its own data on only 13 samples. Paper [6] used transfer learning, where a model trained on one network is reused on another, but it still needed at least 10% shared data and was tested only in simulation.

The newer studies from 2022 - 2024 still leave the same gap. Paper [8] uses only one dataset, paper [7] works across devices but stays inside N-BaloT, paper [9] uses two datasets but tests them separately, and paper [10] checks real device cost but not performance across botnet datasets. So the recent research still does not prove that a model trained in one dataset or network can work well in another [7], [8], [9], [10].

What is needed now is a botnet detection method that is tested across different datasets and real network conditions. It should also work when labelled attack data is limited, because that is closer to real cybersecurity work.

1.4 Research Objectives

General objective: To develop a machine-learning botnet detection method for IoT networks that can work across two datasets with limited labelled data.

Specific objectives:

1. To prepare the Bot-IoT and N-BaloT datasets so they are cleaned, balanced, and ready for training and testing.
2. To design a model that can detect IoT botnet traffic while reducing dependence on one dataset only.
3. To test the model in two settings: same-dataset testing and train-on-one-test-on-the-other testing, using accuracy, precision, recall, F1-score, false alarm rate, and the performance drop between the two settings.
4. To compare the model with baseline methods to see whether it performs better across both datasets.

1.5 Research Scope

This study covers IoT botnet detection using the Bot-IoT and N-BaloT datasets. It focuses on botnet attacks found in these datasets, especially Mirai and BASHLITE. The study will test machine learning and deep learning models in two settings: same-dataset testing and cross-dataset testing. Same-dataset testing means training and testing within the same dataset. Cross-dataset testing means training on one dataset and testing on the other. The results will be measured using accuracy, precision, recall, F1-score, false alarm rate, and the performance drop between the two settings. The work is limited to offline experiments using public datasets.

This study does not cover live deployment on a real network. It will not run the model on real hardware, gateways, or IoT devices. It also does not study encrypted traffic in detail. The study will only detect botnet traffic; it will not block, stop, or respond to attacks after detection. It will also not build a new dataset, because the focus is on testing whether existing public datasets can support cross-dataset botnet detection.

Chapter 2 : Literature Review

2.1 Review of existing studies

These six papers [1] - [6] look at how machine learning can help detect cyberattacks in computer networks, especially in IoT systems. IoT means connected devices such as sensors, cameras, or smart devices. Older security systems usually detect attacks by looking for known attack patterns, but this does not work well for new attacks. The papers test different machine-learning methods, including deep learning [1], [2], [3], [6], decision trees [4], and transfer learning [6]. Most of them report very high accuracy. However, they still have important limits. Many only test the model on one dataset, and they do not clearly show that the same method would work in a different real network.

Popoola et al. (2021) - SMOTE-DRNN [1]

Summary

The problem is rare attack classes have too few examples, so the model just misses them. They used SMOTE first so rare classes had enough examples to train on. They used SMOTE with $k = 3$, so it created synthetic samples for the rare attack classes and made the data more balanced. They used the Bot-IoT dataset, 11 classes with 5 of them rare, and trained a Deep Recurrent Neural Network. The model had 1 recurrent layer, 4 dense layers and a softmax output layer, and it used 37 of the 43 available features. They trained it with Adam at a 0.0001 learning rate and a batch size of 64, with a 60/20/20 split and min-max normalisation over 10 epochs. They got 100% accuracy, 99.50% precision, 99.75% recall, 99.62% F1, 99.87% AUC and 0% false positive rate. One minority class improved from 77.45% to 99.02% recall, and a Friedman test ranked the model first out of 15 models.

Critical analysis

It ran a significance test so it is the only paper that checked if its gain was real. One class went from 0% to 100% recall, but the class only had one test record so the 100% recall means one lucky guess. The Nemenyi test said the recall gains were not significant. SMOTE also nearly doubled training time (1147s vs 631s). They only used the Bot-IoT dataset, so the model was never tested on a different dataset or network to show that it can work in other environments.

Connection / conflict

Paper 1 adds synthetic data and Paper 3 removes real data, so they fix imbalance in opposite ways. Paper 1 tested whether its fix worked but Paper 3 never did, and Paper 1's own test came back not significant, so neither paper really proves its fix works.

Idrissi et al. (2021) - BotIDS (CNN) [2]

Summary

They wanted to compare deep learning models for IoT botnet detection. They proposed a CNN model that they call BotIDS and compared it against models like SimpleRNN, LSTM and GRU. They also use 11 classifications, for 10 attacks plus normal, not just attack versus normal like Paper 3. They used Bot-IoT reduced to 5%, which is 3.6 million records. They prepared the data using one-hot encoding and unit-norm scaling, and they reshaped it into image form so the CNN could read it. They also tried a different version with only the 10 best features so they can get the most out of it. They used a 75/25 split with the Adam optimiser and 10 to 40 epochs. CNN won with 0.99935 accuracy and 0.34 ms inference. CNN beat the other three on both accuracy and speed, so it is the best for IoT detection to sort traffic correctly.

Critical analysis

They tested different models, not just one, so it is a real comparison between models and real testing for accurate detection. They only reported accuracy, but the rarest attack class had just 6 samples out of 3.6 million records, and they did nothing to fix this imbalance, such as SMOTE or undersampling. They used the validation set as the test set, so the results look better than they really are. They also used only 5% of the Bot-IoT dataset, so the results may not work on a different network. The full-feature model did beat the 10-feature version, but this also shows that cutting the features did not improve the model.

Connection / conflict

Paper 1 reported seven metrics and Paper 3 reported five on the same dataset, while Paper 2 reported only two, which makes Paper 2 the highest accuracy but the weakest evidence.

Jithu et al. (2021) - DNN-based IDS [3]

Summary

The problem here is that IoT devices have limited resources, and since there are many new botnet variants that can happen in IoT attacks, this cannot be handled because of these resource constraints, and also because new cases come up that were never seen before. The authors propose an anomaly-based intrusion detection system that is very light and that can use deep learning to separate suspicious traffic from normal traffic. They use Bot-IoT DDoS data with encoding, normalisation and 1:7 rebalancing. They created two hidden layers, each with its own nodes, one with 20 nodes and the other with 12, and the classification is binary: normal traffic is 0 and the attack is 1, and it was trained for 20 epochs. They got 0.95 accuracy, 0.97 recall and

0.98 AUC. They also state that conventional machine learning is not cost-efficient, while the deep neural network is better and more cost-efficient in this case, since IoT networks keep growing nonstop.

Critical analysis

The strength of this paper is that the model is small, so it can work on limited devices. The main weakness is that the test set is only 321 records, which is small, so a few mistakes change the percentage a lot and the 0.95 accuracy could be luck. Bot-IoT is a good dataset, but they only used the DDoS part, and they only tested DDoS so other botnet attacks are not covered. They also claim they beat conventional machine learning, but they need to show the evidence for how it is beating it, and no evidence means the claim is not supported.

Connection / conflict

Paper 1 and Paper 3 face the same class-imbalance problem but solve it in opposite ways: Paper 1 makes synthetic data with SMOTE, while Paper 3 deletes real data by undersampling to 1:7. The synthetic samples may not reflect real traffic, and deleting real data may remove information that the model could take advantage of and learn from.

Abu Al-Haija & Al-Dala'ien (2022) - ELBA-IoT [4]

Summary

They wanted to detect Mirai and Bashlite attacks, but keep the model fast enough to work in real time. They used simple decision-tree models, not deep learning, and tested them on the N-Balot dataset with 611,359 normal records and 7,737 botnet records from 5 devices using MATLAB. They used three simple decision-tree methods: AdaBoost, RUSBoost and Bagged Trees. They used Pearson correlation for feature selection, which helped them choose the most useful data, and Z-score normalisation to put the values on a similar scale. They split the data into 70% training, 10% validation and 20% testing. They also used 5-fold cross-validation and Bayesian optimization to test the model properly and choose its best settings. They got 99.6% accuracy, 98.4% precision, 97.1% recall, and 97.7% F1 for the 10-class test, and they got 100% for the binary and 3-class tests. The model processed about 25,000 samples per second, which is about 40 μ s per sample.

Critical analysis

The good part is that they got very high accuracy and very fast results without using deep learning. The main problem is that they used only one dataset and two botnet families, and the data had a 79:1 imbalance that was not properly fixed. They also tested it on a PC, not on a real IoT device. They did not report AUC, training time or memory use, so it is difficult to fully judge the model's performance and efficiency. They also compared their results with studies that used different datasets, so the comparison may not be fair.

Connection / conflict

Papers 1 - 3 used deep learning, but this paper used simple trees and still got similar results, so it was faster and easier to run.

Nguyen et al. (2022) - collaborative early detection [5]

Summary

They wanted to detect IoT botnet malware early, before it finishes running. Papers 1 - 4 inspect network traffic after infection, but this paper runs the malware inside a sandbox and watches its behaviour while it is still executing. They used dynamic behavioural analysis in V-Sandbox, so the malware was allowed to run while the system recorded what it did. Three models then voted together: KNN for network traffic, KNN for system calls and Random Forest for device resource use. The system calls were changed into a graph and processed using Graph2Vec. The model could decide after 300 system calls, 20 resource behaviours or 50 network packets. They built their own dataset with 8,911 samples: 5,023 botnets and 3,888 benign samples from Mirai and Bashlite. They got 99.37% accuracy, 0.9896 AUC and a 0.0194 false-positive rate. The malware F1-score was 0.9957 and the benign F1-score was 0.9884. Prediction took about 6 seconds.

Critical analysis

The main strength is that it combines system calls, network traffic and device resource use, so it can detect malware from more than one type of behaviour. Although prediction took only about 6 seconds, collecting each sample took around 3 minutes because of the virtual-machine startup and sandbox process, so the claim of early detection is weaker. The dataset was also private, the external test used only 13 samples and the decision thresholds were never checked outside their own sandbox.

Connection / conflict

Papers 1 - 4 mainly detect botnet traffic after infection, but this paper tries to catch the malware while it is running. This makes it more preventive, although it is slower and harder to test in a real environment.

Khoa et al. (2023) - deep transfer learning [6]

Summary

They wanted a network with no labelled attack data to learn from another network that already had some. This directly challenges the usual assumption that every network has enough labelled data for training. They combined federated learning with transfer learning. Network A trained an autoencoder on its labelled data, then shared the learned model weights with Network B. They built a joint feature space so networks with different features could still share knowledge. They also used about 10% mutual samples and an alignment loss to reduce negative transfer, which happens when knowledge from one network makes the other network perform worse. Each network then kept its own fine-tuned model. They tested the method on 12 datasets, including 9

IoT datasets and 3 classical datasets such as N-BalIoT, KDD, NSL-KDD and UNSW-NB15. Accuracy reached about 99%, with more than 40% improvement over the baseline. For example, IoT1 improved from 45.8% to 85.8%, while KDD improved from 80.5% to 99.3%. The model needed at least 10% mutual data and about 200 training epochs.

Critical analysis

This was the only paper that tested across several datasets and directly handled the problem of having little or no labelled data in the target network. They compared the method with only one weak unsupervised baseline and reported only accuracy and AUC, with no precision, recall or F1-score. They also did not give the exact hyperparameter values, tested only two networks and never deployed the model on live traffic.

Connection / conflict

The other papers assume labelled data is already available, but this paper asks how one network can learn from another. This makes it the strongest match for the research gap, although it still needs some shared data and was tested only on static datasets.

Summary Table of Reviewed Papers

#	Year	Author(s)	Methodology	Key findings	Weaknesses / limitations
1	2021	Popoola et al.	SMOTE (k = 3) to create synthetic samples for rare classes, then a Deep Recurrent Neural Network (1 recurrent layer, 4 dense layers, softmax); Bot-IoT, 11 classes, 37 of 43 features, 60/20/20, Adam 0.0001, 10 epochs	100% accuracy, 99.50% precision, 99.75% recall, 99.62% F1, 99.87% AUC, 0% FPR; ranked first of 15 models by a Friedman test; one minority class rose from 77.45% to 99.02% recall	One class had only one test record, so its 100% recall means one lucky guess; the Nemenyi test said the recall gains were not significant; SMOTE nearly doubled training time (1147 s vs 631 s); only Bot-IoT, so never tested on another network
2	2021	Idrissi et al.	CNN (BotIDS) compared against SimpleRNN, LSTM and GRU; Bot-IoT reduced to 5% (3.6 million records); one-hot encoding and unit-norm scaling, reshaped into image form; 75/25 split, Adam, 10 - 40 epochs, 11 classes	CNN won with 0.99935 accuracy and 0.34 ms inference, beating the other three on both accuracy and speed; the full-feature model beat the 10-feature version	Only accuracy reported, so missed rare attacks can be hidden; the rarest class had just 6 samples in 3.6 million records and the imbalance was never fixed; the validation set was used as the test set; only 5% of Bot-IoT was used
3	2021	Jithu et al.	Anomaly-based IDS using a lightweight deep neural network; two hidden layers of 20 and 12 nodes; Bot-IoT DDoS data with encoding, normalisation and 1:7 rebalancing; binary classification, 20 epochs	0.95 accuracy, 0.97 recall, 0.98 AUC; conventional machine learning is not cost-efficient while the DNN is, since IoT networks keep growing	The test set is only 321 records, so a few mistakes change the percentage a lot; only DDoS was tested, so other botnet attacks are not covered; no evidence was shown for the claim that it beats

					conventional machine learning
4	2022	Abu Al-Haija & Al-Dala'ien	Simple decision-tree models (AdaBoost, RUSBoost, Bagged Trees), not deep learning; N-BaloT with 611,359 normal and 7,737 botnet records from 5 devices; Pearson correlation feature selection, Z-score normalisation, 70/10/20 split, 5-fold cross-validation, Bayesian optimisation, MATLAB	99.6% accuracy, 98.4% precision, 97.1% recall, 97.7% F1 on the 10-class test; 100% on the binary and 3-class tests; about 25,000 samples per second ($\approx 40 \mu s$ per sample)	Only one dataset and two botnet families; a 79:1 imbalance that was not properly fixed; tested on a PC, not a real IoT device; no AUC, training time or memory reported; compared against studies that used different datasets, so the comparison may not be fair
5	2022	Nguyen et al.	Dynamic behavioural analysis in V-Sandbox, the malware runs while the system records what it does; three models vote together (KNN for network traffic, KNN for system calls, Random Forest for device resource use); system calls converted to a graph and processed with Graph2Vec; decides after 300 system calls, 20 resource behaviours or 50 packets; self-built dataset of 8,911 samples	99.37% accuracy, 0.9896 AUC, 0.0194 false-positive rate; malware F1 0.9957 and benign F1 0.9884; prediction took about 6 seconds	Collecting each sample took around 3 minutes because of virtual-machine startup and the sandbox process, so the early-detection claim is weaker; the dataset was private; the external test used only 13 samples; the thresholds were never checked outside their own sandbox
6	2023	Khoa et al.	Federated learning combined with transfer learning; Network A trains an autoencoder on labelled data then shares the weights with Network B; a joint feature space lets networks with different features share knowledge; about 10% mutual samples and an alignment loss reduce negative transfer; each network keeps its own fine-tuned model	Tested on 12 datasets (9 IoT, 3 classical: N-BaloT, KDD, NSL-KDD, UNSW-NB15); about 99% accuracy with more than 40% improvement over the baseline (IoT1 45.8% $\rightarrow$ 85.8%, KDD 80.5% $\rightarrow$ 99.3%); needs at least 10% mutual data and about 200 epochs	Compared against only one weak unsupervised baseline; reported only accuracy and AUC, with no precision, recall or F1; exact hyperparameter values not given; only two networks tested; never deployed on live traffic

Synthesis of the Six Core Studies

All six papers [1] - [6] agree on one main point: machine learning is useful for detecting cyberattacks. They all report accuracy above 95%, which makes the results look strong. They also all suggest that machine learning is better than traditional signature-based detection. Signature-based detection means the system looks for attacks it already knows. The problem is that new attacks may not match old patterns. Machine learning is different because it can learn from examples of network traffic and find patterns that may show an attack.

The papers disagree about what type of machine learning is best. Popoola et al. [1], Idrissi et al. [2], Jithu et al. [3] and Khoa et al. [6] use deep learning, which is usually more complex. However, Abu Al-Haija and Al-Dala'ien [4] use simpler tree-based models and still get similar results. This shows that a more complicated model is not always clearly better. The papers also deal with unbalanced data in different ways. Unbalanced data means there are many examples of normal traffic but fewer examples of attacks. Some papers use SMOTE [1], some use undersampling [3], and some do not clearly solve this problem [2], [4].

The main weakness across the papers is that they do not fully prove their methods will work in real situations. Many papers test only one dataset [1] - [4], so we do not know if the model would work on another network. Some papers also do not include enough measurements, such as false alarms or missed attacks [2], [6]. Another issue is that most papers assume labelled data already exists [1] - [5]. Labelled data means the examples are already marked as normal or attack. In real networks, this data may be hard to get, and only Khoa et al. [6] address this directly. Some studies also use simulation instead of real devices [4], [6].

The gap is that these papers show high accuracy in controlled tests, but they do not prove a method that works well across different real networks with limited labelled data.

2.2 Recent studies (2022-2024)

New study 1 : Saied, Guirguis and Madbouly [8] study IoT botnet detection, where the model needs to be accurate but also light enough for IoT use. They compare six tree-based models: Decision Tree, Random Forest, Bagging Meta Classifier, AdaBoost, Gradient Descent Boosting, and XGBoost. The paper tests them in two ways: detection/classification performance and computational performance. The dataset is N-BalIoT, with 1,667,796 balanced instances, 115 features, and three classes: Benign, Mirai, and Gafgyt. Random Forest gives the best result, with accuracy of 0.999991. It also has a low detection time. Compared with the CNN baseline, Random Forest has detection time of 1.298×10^{-5} seconds, while CNN has 10.186×10^{-5} seconds. A strength of this paper is that it checks both accuracy and time cost, so the model is judged more practically. The weakness is that it still uses only one dataset. This means the paper does not prove that the model would work on another network, which is the limitation my work builds on.

New study 2 : Ali et al. [9] study IoT botnet detection, especially for known and zero-day attacks where older rule-based or signature-based systems may fail. They propose a hybrid deep learning model called LAE-MLP. The LSTM part learns sequence and time patterns in network traffic. The autoencoder compresses those patterns into useful features, and the MLP uses those features to classify traffic as normal or attack. The paper tests the model on two datasets. N-BalIoT2018 has over 70.6 million instances, 115 features, and normal traffic plus 10 attack classes. UNSW-NB15 has 2,540,044 instances, 49 features, and normal traffic plus 9 attack classes. The model reports 99.77% accuracy on N-BalIoT2018 and 99.67% accuracy on UNSW-NB15. A strength of this paper is that it uses two datasets and compares the hybrid model with AE, LSTM, and MLP. The weakness is that the datasets are still handled separately, and the

data preparation is not explained clearly enough. The limitation my work builds on is that it does not test whether a model trained on N-BaIoT can work on UNSW-NB15, or whether the reverse works.

New study 3 : Catillo, Pecchia and Villano [7] study IoT botnet detection where many existing IDS methods train a separate model for each device or attack. That is hard to scale in real IoT networks, so they propose one semi-supervised deep autoencoder. The model is trained only on normal traffic, then it flags traffic as an attack when the reconstruction error is too high. It is "all-in-one" because one autoencoder is trained using normal traffic pooled from all devices instead of one model per device. The paper uses the N-BaIoT dataset with nine IoT devices, including doorbells, a thermostat, a baby monitor, a webcam, and security cameras. The attack families are Mirai and BASHLITE, and the task is binary: normal or attack. Table 7 reports recall of 0.9994 - 0.9997, precision of 0.9999 - 1.0, FPR of 0.0 - 0.0071, and F1-score of 0.9996 - 0.9998. A strength is that it tests generalization across devices and reduces the need for many separate models. The weakness is that it still uses only N-BaIoT. The limitation my work builds on is that it does not test outside N-BaIoT or prove cross-dataset performance.

New study 4 : Kumar et al. [10] study early IoT botnet detection at the network edge. Their goal is to detect bots before they are used for attacks, during the scanning/infection and bot-CnC communication stage. They propose EDIMA, which has two stages. First, it uses machine learning on aggregate gateway traffic to detect bot scanning activity. Second, if the gateway traffic looks malicious, it uses ACF-based tests on each device's traffic to find the infected device through bot-CnC communication patterns. They built a testbed with IoT and non-IoT devices and collected normal traffic for four weeks. They also used live malware samples, mainly Ioligang and echobot, and checked with IoT-23 and UNSW IoT. Table V shows Random Forest got AC 1.0, PR 1.0, RC 1.0, and F1 1.0. Table VIII reports Raspberry Pi cost: TAD 415.6 secs, TBD 342.4 secs, 12kB storage, and 96.2MB RAM. A strength is that it tests deployment cost, not only accuracy. A weakness is that the main malicious traffic comes from only two useful malware samples and known centralized botnets. The limitation my work builds on is that it still does not prove performance across different IoT botnet datasets.

Summary of recent studies:

Study (author, year)	Method used	Key finding	Limitation / gap you build on
Saied, Guirguis & Madbouly, 2023 [8]	six tree-based models compared	Random Forest best: accuracy 0.999991, low detection time	only one dataset; does not prove the model works on another network
Ali et al., 2024 [9]	hybrid LAE-MLP (LSTM + autoencoder + MLP)	99.77% accuracy on N-BaIoT2018, 99.67% on UNSW-NB15	datasets handled separately; does not test a model trained on N-BaIoT on UNSW-NB15

Catillo, Pecchia & Villano, 2023 [7]	one semi-supervised deep autoencoder, trained on pooled normal traffic	all-in-one model: recall 0.9994 - 0.9997, FPR 0.0 - 0.0071 across nine devices	does not test outside N-BaloT or prove cross-dataset performance
Kumar et al., 2022 [10]	EDIMA: two-stage ML + ACF tests at the edge gateway	Random Forest 1.0 on all metrics; Raspberry Pi cost 415.6 s detection, 96.2 MB RAM	two malware samples, known centralized botnets; does not prove performance across different IoT botnet datasets

2.3 Synthesis : how the reviewed literature relates to my objectives

Most of the papers stay inside one dataset when they test their models [1], [2], [3], [4], [7], [8]. This means the results can look very strong, but they do not show whether the model works in another dataset or network. SO3 addresses this by testing the model in both same-dataset and cross-dataset settings.

The papers also treat rare attacks in different ways. Paper [1] uses SMOTE, paper [3] uses undersampling, and paper [4] does not clearly fix the imbalance problem. SO1 addresses this by preparing the Bot-IoT and N-BaloT datasets in a balanced way before training and testing.

Paper [9] is important because it uses two datasets, N-BaloT and UNSW-NB15. However, it tests them separately, so it still does not show what happens when a model trained on one dataset is tested on another. SO3 addresses this directly by doing the missing cross-dataset test.

Paper [6] tries to deal with transfer learning, where one network learns from another. However, it still needs at least 10% shared or mutual data, so the model is still partly tied to the target network. SO2 addresses this by designing a model that is less dependent on one dataset only.

Paper [8] shows that a simple Random Forest model can get nearly perfect results and low detection time. This matters because a complex model is not automatically better if a simpler model performs just as well. SO4 addresses this by comparing the proposed model against baseline methods such as Random Forest.

These objectives come straight from the gaps found in the reviewed papers.

Chapter 3 : Research Methodology

3.1 Research methodology

3.1.1 Method for SO1

For SO1, the first step is to get the two public datasets used in this study: Bot-IoT and N-BaloT. After downloading them, I would inspect the files and labels to understand which records are normal traffic and which records are botnet attacks. The next step is cleaning. I would remove duplicate records, remove broken or unreadable records, and handle missing values so the model is not trained on bad data.

After cleaning, I would align the features between the two datasets. This is important because Bot-IoT and N-BaloT do not have the exact same feature set. I would keep or build shared traffic-statistic features that both datasets can support, then normalize them to the same scale. Without this step, cross-dataset testing is not possible because the model would see different input columns in each dataset.

Next, I would split the prepared data into training, validation, and test sets. Then I would balance the classes so rare attacks can be learned properly. I would use SMOTE because paper [1] used it for rare attacks, and it helps create a more balanced training set without simply removing too much data. SMOTE is applied only to the training set, while the test data and the cross-dataset test use no synthetic samples, because synthetic copies in the test set could make the results look better than they really are.

SO1 is completed when the cleaned, balanced, and aligned dataset files are ready for the experiments.

3.1.2 Method for SO2

For SO2, I would design and train two model families instead of depending on only one approach. The first would be a tree ensemble model, mainly Random Forest, because paper [8] showed that Random Forest can reach very high accuracy with low detection cost. The second would be a deep learning model, using ideas from the reviewed papers, such as autoencoder-based detection in [7] and LSTM-based learning in [9]. This gives the study both a simpler model and a deeper model to compare.

Both models would use only the shared and aligned features prepared in SO1. This is important because the aim is not to build a model that only understands one dataset's special columns. By using common traffic-statistic features, the models should be less tied to Bot-IoT or N-BaloT alone.

For the limited labelled data issue, I would also try a transfer-learning setting based on the idea in [6]. A model trained on the source dataset could be adapted using a small amount of labelled target data. This would be tested as a possible improvement, not assumed to work in advance.

SO2 is completed when the trained detection models can run on both prepared datasets.

3.1.3 Method for SO3

For SO3, I would evaluate the models in two settings. The first setting is same-dataset testing. In this setting, I would train and test the model within Bot-IoT, then train and test the model within N-BaloT. These results give the reference numbers because the model is tested on data from the same source it learned from.

The second setting is cross-dataset testing. In this setting, I would train the model on Bot-IoT and test it on N-BaloT, then train the model on N-BaloT and test it on Bot-IoT. Both model families from SO2, Random Forest and the deep learning model, would go through both settings.

For every run, I would record accuracy, precision, recall, F1-score, and false alarm rate. The most important value is the performance drop between Setting 1 and Setting 2. This drop answers the research problem directly. If the drop is small, the model is generalising across datasets. If the drop is large, the model may only be learning patterns from its own dataset.

The transfer-learning variant from SO2 would also be tested in the same way, using a small amount of labelled target data. SO3 is completed when the full results table for both settings is filled.

3.1.4 Method for SO4

For SO4, I would compare the proposed shared-feature approach with a baseline that reflects the common method in the literature. The baseline would be a model trained on one dataset using that dataset's own full feature set, then tested only on the same dataset. This is the standard pattern used in many reviewed papers [1], [2], [3], [4], [8].

My comparison would then place the baseline and the shared-feature models under the same cross-dataset test from SO3. This makes the comparison fair because both approaches are judged using the same train-test directions, metrics, and evaluation process.

Success would mean that the shared-feature model has a smaller performance drop than the baseline, or better cross-dataset results overall. Published paper results can be mentioned as background, but they cannot be the main fair comparison because each paper uses different features, data preparation steps, and test settings. The fair result must come from my own experiments under the same conditions.

SO4 is completed when the comparison table is filled with the baseline and proposed model results.

3.2 Expected outcomes

The study is expected to produce three main outputs. First, it should produce cleaned and aligned versions of the Bot-IoT and N-BaIoT datasets. These files will remove duplicate or broken records, handle missing values, balance the classes, and keep shared traffic features that both datasets can support. Second, the study should produce trained detection models. These will include a Random Forest model, a deep learning model, and a transfer-learning variant that uses a small amount of target data. Third, the study should produce filled results tables and a comparison table. These tables will show the same-dataset results, the cross-dataset results, the performance drop, and the comparison between the proposed approach and the baseline.

The results are expected to show very high scores in same-dataset testing, similar to the strong results reported in the reviewed papers. They are also expected to show a real drop in cross-dataset testing, and this drop is the new evidence that the study is trying to measure. The shared features and transfer-learning variant should make the drop smaller because the model is less tied to one dataset only. Even if the drop stays large, that would still be a useful finding because it would show that single-dataset accuracy can overstate real performance.

3.3 Objective → method → outcome trace table

Objective ("To ...")	How Chapter 3 achieves it (method / tool / metric)	Expected outcome
SO1 To prepare the Bot-IoT and N-BaIoT datasets	cleaning, feature alignment, train/validation/test split, SMOTE balancing (training data only) (Section 3.1.1)	cleaned, balanced, and aligned dataset files
SO2 To design a model less dependent on one dataset	Random Forest + deep learning model on shared aligned features, plus transfer-learning variant (Section 3.1.2)	trained detection models that run on both datasets
SO3 To test in two settings	same-dataset and cross-dataset runs; accuracy, precision, recall, F1, false alarm rate; the performance drop (Section 3.1.3)	full results table for both settings
SO4 To compare with baselines	baseline vs shared-feature models under the same cross-dataset test (Section 3.1.4)	comparison table; smaller performance drop than the baseline

3.4 Gantt chart

Task / milestone	W1	W2	W3	W4	W5	W6	W7	W8
Literature review	■	■						
Dataset acquisition & preprocessing		■	■					
Model design / implementation			■	■	■			
Evaluation & comparison					■	■		
Analysis of results						■	■	
Writing up & submission						■	■	■

References

- [1] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "SMOTE-DRNN: A deep learning algorithm for botnet detection in the Internet-of-Things networks," *Sensors*, vol. 21, no. 9, p. 2985, 2021, doi: 10.3390/s21092985.
- [2] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for IoT against botnet attacks," *IAES Int. J. Artif. Intell.*, vol. 10, no. 1, pp. 110-120, 2021, doi: 10.11591/ijai.v10.i1.pp110-120.
- [3] P. Jithu, J. Shareena, A. Ramdas, and A. P. Haripriya, "Intrusion detection system for IoT botnet attacks using deep learning," *SN Comput. Sci.*, vol. 2, no. 3, p. 205, 2021, doi: 10.1007/s42979-021-00516-9.
- [4] Q. Abu Al-Haija and M. Al-Dala'ien, "ELBA-IoT: An ensemble learning model for botnet attack detection in IoT networks," *J. Sens. Actuator Netw.*, vol. 11, no. 1, p. 18, 2022, doi: 10.3390/jsan11010018.
- [5] G. L. Nguyen, B. Dumba, Q.-D. Ngo, H.-V. Le, and T. N. Nguyen, "A collaborative approach to early detection of IoT botnet," *Comput. Electr. Eng.*, vol. 97, p. 107525, 2022, doi: 10.1016/j.compeleceng.2021.107525.
- [6] T. V. Khoa et al., "Deep transfer learning: A novel collaborative learning model for cyberattack detection systems in IoT networks," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8578-8589, 2023, doi: 10.1109/JIOT.2022.3202029.
- [7] M. Catillo, A. Pecchia, and U. Villano, "A deep learning method for lightweight and cross-device IoT botnet detection," *Appl. Sci.*, vol. 13, no. 2, p. 837, 2023, doi: 10.3390/app13020837.
- [8] M. Saied, S. Guirguis, and M. Madbouly, "A comparative analysis of using ensemble trees for botnet detection and classification in IoT," *Sci. Rep.*, vol. 13, p. 21632, 2023, doi: 10.1038/s41598-023-48681-6.

[9] S. Ali *et al.*, "A novel approach of botnet detection using hybrid deep learning for enhancing security in IoT networks," *Alex. Eng. J.*, vol. 103, pp. 88-97, 2024, doi: 10.1016/j.aej.2024.05.113.

[10] A. Kumar, M. Shridhar, S. Swaminathan, and T. J. Lim, "Machine learning-based early detection of IoT botnets using network-edge traffic," *Comput. Secur.*, vol. 117, p. 102693, 2022, doi: 10.1016/j.cose.2022.102693.