A Machine Learning Approach to Detect Anomalies on Edge Devices

MSc Research Project
Data Analytics

Awais Shamas
Student ID: x23258756

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Awais Shamas		
Student ID:	x23258756		
Programme:	MSc. Data Analytics	**Year:**	2024-2025
Module:	MSc. Research Project		
Supervisor:	Mr. Vikas Tomer		
Submission Due Date:	12 December, 2024		
Project Title:	A Machine Learning Approach to Detect Anomalies on Edge Devices		
Word Count:	12473 **Page Count** 34		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……………………………………………………………………………………………………

Date: ……………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Machine Learning Approach to Detect Anomalies on Edge Devices

Awais Shamas
x23258756

Abstract

Over the years, Internet of Things (IoT) devices are increasing rapidly and is expected to extend over 20 billion by 2030. The growth underscores the increasing demand for vigorous security solutions to secure these devices from cyber-attacks. Anomalies in IoT data can lead to failure and unexpected behavior in a system. Therefore, it is crucial to detect anomalies to achieve system performance and reliability. Detecting anomalies in resources constraint devices, such as IoT devices, cause some challenges. In this research, we proposed machine learning based anomaly detection system implemented for edge device such as Raspberry Pi. Our research uses TensorFlow Lite that helps in developing a compressed model that detects malicious activities in real-time resource usage processes without demanding high computational resources. An autoencoder-based model was implemented to detect anomalies in resource usage processes. Models were trained on high performance devices and were further deployed on resource constrained hardware such as Raspberry Pi Zero 2W. The real-time inference happens every 5 seconds indicating highly accurate and timely anomaly detection for both full and compressed models by achieving an accuracy of 97.00% showcasing that lightweight models can outperform full models on the resource overhead. The key contribution of our research is the development of lightweight, scalable model for protecting the fast-growing IoT device ecosystem, which should identify new threats efficiently while assuring effective anomaly detection in resource-constraint environments.

1 Introduction

Internet of Things (IoT) allows various devices to connect and interact with each other and with their surroundings. These devices have capabilities of collecting data and performing tasks automatically. Internet of Things (IoT) is becoming famous due to its innovative concept and ideas (Gharavi, et al. 2024). IoT environments produce massive amount of data that could be helpful to improve decision making process and optimizing systems (Dastjerdi, et al. 2016). IoT provides the facility of real-time monitoring and controlling of the system using devices like computers, smartphones and tablets. Collecting and analyzing data using cloud processing techniques help to make the complex interactions amongst multiple devices. An important application of IoT is in industry 4.0, that focuses on the improvement of industrial processing (Jove, et al 2022). The ability of IoT devices to integrate smart systems across multiple environments offer benefits in many fields including environmental, financial, smart homes, industries and healthcare. One of the most effective applications of IoT is in smart homes where interrelated devices provide comfort, efficiency and convenience. For example, integration of virtual interaction services and entertainment can help to alleviate the feelings of loneliness and isolation. Power monitoring capabilities in smart home can lessen the energy consumption (Mota, et al. 2024) Out of 24.1 billion devices, it is estimated that 5.8 billion would be dedicated to industrial applications and enterprises (Says, et al. 2019). Study shows that currently 12.08 billion IoT devices are linked with each other, and it is expected that these number will increase to 29.04 billion by 2030 (Statista, n.d.).

The widespread adoption of IoT technology caught the attention of cyber attackers and they seek to exploit these systems using advance hacking techniques including botnets. Botnets malwares are used to launch Distributed Denial-of-service (DDoS) attacks with bandwidth reaching up to 1.1 Tbsp. The vulnerability in IoT is further increased due to the lack of standardization and the occurrence of lightweight, inexpensive and low powered devices in IoT networks (Koroniotis, et al. 2019). Security has become a major concern in IoT devices as IoT devices are connected with various networks and there are chances that they will be disposed to more attacks as compared to isolated systems. Antivirus and firewalls help to protect the system, but delicate security policies lead to breaches. There are many low-priced IoT devices that are available in the market. Manufacturers overlook the crucial things like privacy and security in IoT devices that results in the leakage of user's confidential information. Mirai malware, that was discovered in 2016, demonstrates how IoT devices could be hijacked and then used as bots to implement large scale attacks such as Distributed Denial of Service attack (DDoS), potentially destroying healthcare systems and endangering human lives at risks (Antonakakis, et al. 2024).

The advent of IoT has introduced challenges in areas like maintenance, data storage, privacy and security. Many IoT devices are designed in a way so they can perform only specific hardware and software tasks. For example, using fast processor for real time performance, without increasing any extra power in it. These are the factors due to which IoT devices are often launch with the vulnerabilities in it. Once a device is compromised, various malicious activities can be exploited on it (Breitenbacher, et al. 2019). Due to specific applications of IoT devices, their security is limited. For instance, wearable devices, handheld and portable devices, that rely on battery often compromise on security by prioritizing longer battery. To highlight these issues, control protocols and energy efficient communication have been developed to address these issues but vulnerabilities in these protocols still exist. An important example is Bluetooth Low Energy (BLE) communication protocol, that is broadly used in industrial IoT (IIoT) and wearable devices. Study shows that BLE is vulnerable to various attacks. User data is expose due to this protocol as the transmitted packets are sent in the form of plain text and during the process of reconnection between paired devices there are chances of exploitation (Wu, et al. 2020), (Antonioli, et al. 2022).

The major concerns for smart home devices users are security. Aldossari, et al. (2018) research highlights that users are most likely to adopt these technologies based on their trust in the system and how much privacy and security it offers to the user's data (Alyasiri, et al. 2021). Traditionally, there were three main goals of the IT security: confidentiality, integrity and accountability (Schiller, et al. 2022). Due to the online availability of IoT devices, they are more prone to attacks. More security challenges take place due to the embedded technology within IoT devices. For instance, an attack on smart home device would cease the working of refrigerator to a smoke detector that stops operating, putting lives at risk. Attacks on smart home systems result in the leakage of personal information and often leads to serious crimes. For instance, misuse of video feeds can be done by hackers for burglary or temper healthcare devices that could result in causing physical harm. Many other risks take place due to poor security in IoT devices including leakage of personal information, unauthorized access to devices, accidently activating the devices and other malware attacks (Li, et al. 2023). Huge amount of data is generated by IoT devices, but they have limited memory and computing resources. While cloud computing help in overcoming these limitations, but it produces high latency which can cause issues in areas like health monitoring. Therefore, solely depending on cloud solutions is unsuitable in these cases (Dastjerdi, et al. 2016). Edge computing can solve high latency issues and privacy concerns by processing and storing data in the nodes from IoT devices at edge networks. By using this method, data can be preprocessed to remove sensitive information before transferring it to the cloud. However, reliance on large-scale computing resources is quite expensive and the boundaries of the edge networks are not defined clearly (Premsankar, et al. 2018). Some of the most typical approaches are Intrusion Detection Systems in IoT systems that monitor network traffic and reduce the need for high computational power at the device. IDS can be implemented without making any changes to the configuration of the existing IoT systems. IDS has two types: Anomaly base IDS and Signature base IDS. Anomaly based IDS are designed to detect unusual and unknown attacks by monitoring the system for deviations from usual traffic patterns. Signature base IDS depends on pre-defined rules to access the attacks that are connected with well-known behaviors and patterns. This is achieved by doing contrast between unusual traffic against predefined normal patterns by using Artificial Intelligence (AI) models that are trained on the previous historical data (Jove, et al 2022).

Machine learning (ML) is utilized in many fields for years. Development of new machine learning frameworks including Keras, PyTorch and TensorFlow has increased the interest of many researchers in the field of cybersecurity. Huge research has been conducted on detecting anomalies and attacks using machine learning and various algorithms have been introduced that would help in detecting anomalies in IoT devices that shows notable effectiveness in identifying threats. Accuracy of the machine learning models depends a lot on the size of dataset that is used for training and there is often trade-off between model accuracy and its size. This is not a problem for high-end devices with sufficient computing resources to achieve optimal performance and accuracy, but it becomes a challenge for the devices that have limited resources. For example, TensorFlow lite, that Google introduced in 2019, was designed for mobile, embedded and low-end devices to compress large models into lightweight models. This framework compressed the version of the models without affecting its performance and accuracy. To detect anomalous activities, majority of the machine learning detection-based systems are trained on the labeled datasets. Limitation of this approach is that it will not detect latest malicious activity that model is unaware of. For this research, we will highlight this challenge and develop a lightweight, compressed machine learning model. With the help of TensorFlow lite, we will detect the anomalies on edge devices, evaluate the performance and accuracy on resource constraint devices.

Research Question. The above research problem motivates the following research question:

- How effectively can a compressed, lightweight machine learning model detect anomalous activities on resource usage of processes using edge devices?

The objectives of this research are to:

- Develop an anomaly detection model trained on resource usage of processes running on the edge device.
- Convert the model into compressed, lightweight version using TensorFlow Lite.
- Deploy the lightweight model on edge devices such as Raspberry Pi.

Below are the remaining sections of this research:

- **Section 2:** A detailed literature review of the work done in this domain.

- **Section 3:** A detailed methodology to achieve the objectives of this project.
- **Section 4:** Evaluation
- **Section 5:** Discussion, conclusion and future work.

2 Related Work

This section gives an overview of the related studies by methodologically arranging the previous studies which is also shown in the gap analysis Table 1. These studies focus on detecting anomalies on edge devices using machine learning approach. Researchers highlights range of methods for detecting anomalies from traditional machine learning models to advance deep learning models that improves the accuracy but demands high computational power. To overcome these challenges, studies suggested lightweighted models striking balance between performance and efficiency on edge devices.

2.1 Anomaly Detection Techniques in IoT Devices

As the usage of Internet of Things (IoT) devices is growing rapidly, it is crucial to make sure that these devices detect threats and unusual behaviors, also known as anomalies. It is essential because many IoT devices, including wearable or smart home devices have limited memory and processing power. Due to this, it becomes daunting to use complex machine learning model on these devices. Researchers are working on creating faster and lighter models that can detect the anomalies effectively without utilizing many resources.

Huc, et al. (2021) conducted a study to test the performance of different machine learning models on a small and resource-limited device (Raspberry pi 4) to detect the anomaly. Models like Support Vector Machines (SVM), Logistic Regression, Random Forests, Decision Trees and Artificial Neutral Networks (ANN) were implemented on a DS2OS dataset that consists of numerous types of normal and abnormal behaviors. The main goal behind this study was to detect how better models detect anomalies and how much computational power is used by these models. Their study shows that Decision Tree and Random Forest performed better on small datasets by achieving an accuracy score above 95%. SVM requires loads of processing power that does not make it much efficient for small devices. Authors did not discuss about compression and optimization of models leaving it for future work.

Artificial Intelligence plays a vital role in securing IoT networks as complexities are increasing within the network exposing it to security threats (Gudala, et al. 2019) suggested that an unusual behavior in network traffic can be detected with the help of Artificial Intelligence by making a comparison between traditional method with machine learning algorithms. Traditional methods including threshold-based detection is speedy and minimal resources are required in it. However, they cannot adapt well to changings in environment due to which they often produce false alarms. On the other hand, Machine learning models, while resource-intensive, identify more complex attack patterns that are missed by simple methods. Models including SVMs, Random Forests, and Autoencoders were proven to be more efficient and accurate in identifying unknown attacks. AI driven response systems are also discussed in this study, where system automatically respond to the threats in real time, for example, without human intervention, by adjust security setting or by applying patches system automatically address the unknown threats. Despite the benefits of Artificial Intelligence (AI) in IoT network, researchers have pointed out that AI models are much resource-hungry to be deployed on IoT devices which have limited processing power and memory. Researchers suggested that future research should focus on optimizing AI models and creating lightweight models for these devices. Researchers have also emphasized on Federated Learning model, where multiple devices cooperate to train a shared model without sharing any personal information.

Selecting the best model for detecting anomaly in IoT networks is not straightforward because each model has its own strengthens and weaknesses depending upon the dataset and the type of attacks on IoT networks. (Inuwa, et al. 2024) did comparison of several machine learning models, including Artificial Neural Networks (ANN), K-Nearest Neighbors (K-NN), SVM, Logistic Regression (LR), and Decision Trees (DT) using the ToN-IoT and Bot-IoT datasets that simulate real world attack scenarios including Denial of service (DoS) and Distributed Denial of Service (DDoS).The results showed that Artificial Neural Networks (ANN) models had the best performance in terms of accuracy and achieved near-perfect scores for certain datasets. Logistic Regression underperformed in terms of complex attacks. Random Forests and Decision Trees achieved great accuracy but they faced challenges with more complex attack patterns including Man-in-the-middle attacks. One limitation of this study includes not focusing on the lightweight version of these models which would be fruitful for deployment on the devices with confined resources. It aligns with the ongoing research of creating lightweight models that can perform well in resource constraint environments like IoT devices.

Chatterjee, et al. (2022) examined numerous anomaly detection techniques and their uses in IoT system. They examined 64 research papers and categorized the methods into four categories including machine learning models, geometric methods, statistical approaches, and deep learning architectures. Their study showed that machine learning and deep learning models are better at detecting complicated anomalies compared to statistical methods. They also observed that mostly machine learning and deep learning models are very demanding in terms of processing power. Researchers emphasized on the lightweight models, including TensorFlow Lite and recommended hybrid models, involving multiple techniques might result in better balance between resource usage and accuracy. Their study is beneficial for the present research, that has an aim of developing models that work effectively in real time on edge devices without wasting much resources.

Haji, et al. (2021) discussed about the difficulties that occur when applying machine learning in detecting anomalies in IoT networks. They grouped multiple models, from simpler ones including Decision Tree and Logistic Regression to more sophisticated models including Recurrent Neural Network (RNN) and Deep Neural Network (DNN). They examined that basic models work well for small datasets and pattern attacks but for complex pattern attacks such as Denial of Service (DoS) or Distributed Denial of Service (DDoS) demands more machine learning power. However, deploying these machine learning models on IoT networks is daunting due to memory and computational power. Authors suggested to explore more lightweight models approach including pruning i.e., removing unnecessary parts of the model and quantization (eliminating the precision of the numbers the model uses) to make deep learning models more compatible for IoT devices. Their study underlines the significance of finding stability between detection accuracy and computational accuracy. It is an important focus of the current research which aims to develop compressed models that can work efficiently in detecting anomalies without overloading resource constraint devices.

2.2 Machine Learning Models on Edge Devices

Murshed, et al. (2021) gave a comprehensive analysis on deploying machine learning (ML) models on edge devices, that are small computers or sensors that are located near where data is collected. Authors address how machine learning (ML) models can be optimized in a way so that they can work in surroundings having limited resources, including battery life, processing power and low memory. In traditional setups, data that is collected from the edge devices were send to the cloud for further processing but this method causes latency issues along privacy concerns. Rather, processing data locally on edge devices can lessen these issues. However, edge devices struggle to work with full-size machine learning models so it is crucial to have lightweight versions of these models.The study pointed out on essential approaches to optimize these models for edge devices, including pruning i.e., removing unnecessary parts of the model, reduction in the size of the model through quantization i.e., use of fewer bits of representing data, and knowledge distillation i.e., using a simple model that is trained by a larger model. These methods help in making models work efficiently without compromising much on accuracy. Some advance lightweight architectures referred to SqueezeNet, MobileNet, and ShuffleNet. These models are designed in a way that help in reducing computational cost and memory usage. For instance, ShuttleNet and MobileNet use separable convolutions depthwise, i.e., a type of layer that reduces the size of the model and make calculations quicker. Meanwhile, SqueezeNet minimizes the number of parameters i.e., the parts of model that are learned during the training phase. SqeezeNet lower the number of parameters by 50 times as compare to the popular model, AlexNet model. These enhancements are important for real-time applications such as autonomous vehicles, video analytics. However, difficulties remain, as speed, energy consumption, balancing accuracy is still daunting, especially in resource-constrained environments. Authors also highlight the concept of Federated Learning, where data from multiple edge devices is gathered for training a model without sending data to the central server. It is beneficial to maintain privacy but becomes a challenge of managing communication between devices. Study suggests of developing collaborative systems where cloud and edge processing work together to achieve best accuracy in terms of efficiency and accuracy.

Fanariotis, et al. (2023) study focused on the power efficiency of machine learning models that run on small IoT devices (Internet of Things), like STM32H7 and ESP32, that are commonly use in edge computing. While previous researchers focused on making machine learning models faster and smaller, this study took a different angle by focusing on how much power these machine learning models can use. It is crucial consideration for devices that often run on batteries. Researchers test both compressed and uncompressed versions of models, including MobileNet-025 and LeNet-5. Compression techniques help in reducing the size and power consumptions of the models without compromising much on accuracy. For example, the compressed version of LeNet-5 consumed 5 times less power than the original model, only decreasing 2% of accuracy. Similarly, when MobileNet-025 is compressed, its power efficiency increased by 3% making it work better for low-poor devices. However, study also highlighted some issues. For example, older versions of TensorFlow were not completely compatible with some certain devices and some compressed models still struggle with low memory capacities. Authors also suggest on focusing more on response times and power efficiency especially for real time tasks.

Tekin, et al. (2023) investigated the performance of different machine learning models when they are used for detecting intrusions i.e., hacking attempts in IoT networks. These networks are composed of small home devices like cameras and thermostats, which could be a target for cyber attackers. Authors composed several models, including k-Nearest Neighbor (k-NN), Logistic Regression, Naïve Bayes (NB), Random Forest (RF), Artificial Neural Network (ANN) and Decision Tree (DT). They focused on two major factors, power consumption of each model and accuracy of each model at detecting intrusions. K-NN and ANN models proved to be more accurate, but they consumed excessive power to be practical for real time use on edge devices. On the other hand, Decision tree and Random Forest displayed better results by offering high accuracy and low energy consumption. Researchers also highlighted about TinyML that helps in optimization of the models without compromising much accuracy. It is crucial for intrusion detection in small homes.

Table 1: Summary of literature review

Author (Year)	Dataset	Methodology	Model	Gaps Identified	Limitations
Huc et al., 2021	DS2OS	Five machine learning models tested and evaluated on Raspberry Pi.	Logistic Regression, SVM, Random Forest, Decision Tree, ANN	1-The dataset doesn't have any features of the resource. 2-Neither Compressed model was used.	1-High resource usage needed for SVM. 2-Poor performance for Logistic Regression on small scale-based data.
Gudala et al., 2019	IoT traffic dataset	Discussed AI techniques for anomaly detection and explored supervised and unsupervised learning specifically on network data.	SVM, Random Forest, Autoencoder	1-The main focus was on network data. 2- It also lacks resource usage processes data. 3- For edge devices there were no lightweight models.	1-No analysis of TensorFlow Lite or other lightweight deployment strategies for resource constrained devices.
Skaperas et al., 2024	Synthetic ARMA	The algorithms were tested and evaluated on cloud systems.	CUSUM and BOCPD	1-It lacks a real time dataset. 2-No process level anomalies detection. 3-Lack resource usage of edge devices	1- No lightweight model deployment on edge devices.
Inuwa et al., 2024	ToN-IoT, BoT-IoT	Comparative Analysis of five supervised models for the detection of the data.	SVM, ANN, DT, LR, K-NN	1-Dataset focused on IoT network data. 2-Lacks resource level usage processes.	1- Poor performance of LR. 2- High resource demand for ANN and SVM

				3-No lightweight deployment model.	
Chatterjee et al., 2022	N/A	Multiple classifications of IoT anomaly detection methods and applications.	Statistic AI methods, SVM, LSTM, AE	1-Lack of exploration into lightweight models like TensorFlow Lite and on-device optimizations	1-High computational cost for ML/DL models in real-time IoT settings.
Haji et al., 2021	N/A	The ML models for anomaly detection in IoT were comparatively reviewed.	SVM, LR, DT, DNN, RNN	1-Lack of exploration and investigation into lightweight models like TensorFlow Lite and optimized deep learning for constrained devices.	1-High computational overhead and memory usage for deep learning models
Murshed et al., 2021	N/A	Survey of ML models and techniques for edge deployment	MobileNet, ShuffleNet, SqueezeNet	1-Lack of hybrid optimization strategies to balance accuracy and latency, and no consideration of TensorFlow Lite or quantized versions to further reduce resource consumption on constrained devices.	1-Communication overheads in federated learning 2-Limited scalability for large networks.
Tekin et al., 2023	DS2OS	Energy consumption	LR, k-NN, DT, RF, NB,	1-No exploration	1-High energy consumption

		inspection and analysis for three different training and two inference approaches	ANN	of TensorFlow Lite or energy-efficient deployment techniques for edge-based IoT networks.	for k-NN and ANN in real-time applications.
Nkuba et al., 2023	Real-world Z-Wave traffic data (Data collected from 17 different IoT devices)	Packet formalization, centralized learning	ANN	1-No exploration of TensorFlow Lite or on-device training techniques to handle real-time updates and adaptations.	1-Lack of real-time handling of newly added devices requires retraining and reequipping for network updates
Zhang et al., 2022	GANomaly, ResNet-18, and other image/video datasets	Block-grained scaling mechanism.	LightDNN (custom DNN framework)	1-Lack of dynamic adaptation to new scenarios without pre-generated block combinations.	1-Requires some degree of offline profiling and training for initial block generation.
Wang et al., 2022	HDFS, BGL	Low-dimensional semantic vectors, multi-kernel pointwise convolution	Lightweight TCN	1-Did not explore TensorFlow Lite for optimizing temporal 2-convolutional networks or hybrid architectures for real-time log anomaly detection.	1-Limited handling of real-time variations in log templates
Yap et al., 2021	N/A	Review of TinyML techniques and model compression	Pruning, Quantization, Knowledge Distillation, TensorFlow	1-No exploration of real-time on-device training and	1-Loss of accuracy in heavily compressed models,limited

		for anomaly detection	Lite Micro	adaptation strategies to handle evolving patterns in constrained devices.	scalability for complex anomaly patterns
Ziegler et al., 2023	MVTec AD	On-device optimization, quantization, and model conversion for MCUs	MCU-PatchCore (based on PatchCore and MCUNet)	1-No consideration for handling new unseen anomalies dynamically without model re-deployment.	1-Lack of real-time retraining, limited adaptability to evolving anomaly patterns
Idrissi et al., 2022	MQTT-IOT-IDS2020	Model compression (post-training quantization, pruning, clustering)	DL-HIDS (optimized CNN)	1-No focus on integrating dynamic model updates or on-device retraining strategies for evolving threats.	1-Unable to deploy on low-resource devices like Arduino Uno due to memory constraints
Antonini et al., 2023	Simulated data in extreme industrial environment s	Tiny-ML Ops, blockchain for logging	Isolation Forest	1-Lack of real-time retraining or dynamic updates to adapt to evolving anomalies without any external support made.	1-Limited model adaptability to new patterns; manual arbitration needed.

2.3 Lightweight Models for Anomaly Detection

Nkuba, et al. (2023) introduced ZMAD to detect anomalies in smart home devices and uses Z-wave protocol, a method to communicate in smart home automation. To remove security flaws ZMAD uses lightweight Artificial Neural Networks (ANN) in older S2 and Z-wave devices. By analyzing network traffic, it identifies the threats and isolates the data that indicate to unexpected behavior. It helps in reducing the complexity of the data by separating irrelevant information that helps in more effective and precise threat detection. Researchers tested ZMAD on 17 real world Z-Wave devices and achieved detection accuracy of 98% while sharply shrinking the model size by up to 47 times as compare to other more complicated systems like Recurrent Neural Networks (RNNs) and Long Short-Term memory (LSTM). ZMAD played a vital role in the detection of numerous types of cyberattacks, such as remote code injection, Denial of Service (DoS) attack and attacks that manipulate the route of network traffic. Researchers notice that multiple challenges are faced by ZMAD when it detects attack on recently added devices without having any need to retrain the whole model. This study is relevant to the current study as it focuses on the usage of lightweight models for detecting anomalies in IoT networks. ZMAD provides an efficient way to recognize suspicious activities in structures communication systems like Z-wave by streamlining data analysis process and applying centralized learning.

Zhang, et al. (2022) proposed LightDNN, which is a framework designed to handle long processing time and high computational demands that usually accompany complex Deep Neural Networks (DNN) for anomaly detection. LightDNN breaks a larger neural network into small independent parts known as blocks. Every part can be compressed and optimized independently. It also reduces computation power and time needed to train and make predictions. It assists the model to adjust the size and accuracy with available computing power provided at any point of time. During experiments on devices like NVIDIA Jetson TX2 and Raspberry pi 4, researchers found out that anomaly detection accuracy can be improved up to 17.4% by using LightDNN while keeping the number of resources same. The ability of LightDNN to scale itself according to available resources enables it to meet the requirements of the real-time applications while minimizing energy consumption. Despite having advancements, LightDNN have some drawbacks. To create initial block combinations, system requires offline training and profiling which makes it less flexible in fast-changing environments. Researchers suggest that future work should focus more model flexibility, that will enable model to adapt quickly in changings scenarios without having need of lengthy retraining periods. This study is relevant to the present study, which focus is on development of efficient and scalable anomaly detection models for edge devices. Block-level optimization strategy of LightDNN provides firm base to make anomaly detection models more versatile to different resource constraint environments.

Wang, et al. (2022) introduced LightLog, a system that is design to detect anomalies in logs generated by computer and other devices. It is crucial for real-time monitoring system to detect potential issues rapidly. The issue with the existing systems, such as LogAnomaly and DeepLog is that it takes a lot of computing power and can be slow. LightLog highlights two main techniques to make the processor efficient and faster. First, LightLog creates low-dimensional semantic vector space. In simpler words, it condenses log data into much smaller and more meaningful form using a post-processing algorithm and word2vec. This helps in reducing the data size by 98%, making it easier to analyze. Second, LightLog applies lightweight version of temporal computational network (TNN), a time of neural network that is design for analyzing data over time. To classify log anomalies, TCN uses techniques like global average pooling i.e., a way for summarizing information and multi-kernel pointwise convolution i.e., a method of speeding up computation process. Testing on two datasets (HDFS and BGL), LightLog outperformed other models by achieving F1-score of 97.0% and 97.2% respectively. Despite these strong outcomes, study highlights some challenges, especially when model has to handle the new types of logs that are introduced during real-time operations. Researchers suggested that future research should focus more on making the model more capable and flexible of learning from new data on the fly, without any need of extensive training. This study aligns well with the current goal of the study of creating efficient anomaly detection models for edge devices. To analyze log data in resource limited environments, LightLogs provides a helpful approach by reduction in computational cost and enhancement in TCN structure.

2.4 TinyML Models for IoT Devices

Yap, et al. (2021) made an analysis on how Tiny Machine Learning (TinyML) can detect anomalies in devices like microcontroller units (MCUs) as they have limited resources. TinyML is a rapidly growing field that combines machine learning models into small, low-powered devices that would be helpful to detect anomalies in IoT networks. These small devices take placed at edge networks and contain finite power and memory. It is crucial that those machine learning models that run on them should be small and efficient. Study highlights techniques including model compression, that involves methods like quantization, knowledge distillation and pruning. These models help in compressing the size and energy consumption of machine learning models, making them adjustable with the limitations of MCUs. Quantization reduces numerical precisions and simplifies calculation, pruning removes unnecessary parts of the model, and knowledge distillation delivers knowledge from a larger model to the lower model while maintaining most of the accuracy. Traditional machine learning methods like Support Vector Machine (SVM) and Decision Tree (DT) function well in these areas but they are not much flexible and accurate as compare to deep learning models that can detect complex patterns. To fill this void, authors discuss tools like Single Value Decomposition (SVD) and TensorFlow Lite Micro which is helpful in deploying compressed versions of deep learning models, such as Convolutional Neural Networks (CNNs) on MCUs. Authors also highlighted that compressing models too much result in reduction of accuracy, which means some crucial anomaly patterns might go unnoticed. To overcome this issue, they suggested that future work should focus on hybrid models that can dynamically adjust the computing power based on real-time conditions like availability of power and usage needs. This review is relevant to the current study as it explores how to optimize machine learning models to be both effective and efficient for real-time anomaly detection on low-power devices.

Ziegler, et al. (2023) proposed MCU-PatchCore, a system that integrates a powerful anomaly detection method called PatchCore with the lightweight architecture known as MCUNet, which is particularly design for low-power MCUs. PatchCore demands the power of graphical processing units (GPUs) to operate but Ziegler adapted to operate it on devices with limited processing power and memory,specifically MCUs with less than 1MB of Flash Storage and 200KB of SRAM. For anomaly detection, MCU-Patchcore was detected on a standard dataset, achieving an accuracy of 86%, which is quite close to the accuracy achieved by systems having powerful hardware. This study highlights several innovations, such as a procedure of converting complex neural networks into light weight versions using TensorFlow Lite (TFLite) and decrease memory usage through quantized operations. However, a limitation of MCU-PatchCore is its lack of ability to handle real-time model retraining and adjustment in new types of anomalies without manual intervention which makes it less flexible in fast-growing environments like industrial setting. Ziegler suggested that future researchers can work on developing models that could retrain themselves on-device or for more dynamic learning, they can collaborate with cloud systems. This study is relevant to the current research as it shows how robust models can be adapted to utilize on low-power devices, aligning with the goal of building scalable and efficient systems for anomaly detection for IoT devices.

Idrissi, et al. (2022) introduced a Deep Learning-Based Host Intrusion Detection System (DL-HIDS) optimized for low power IoT edge devices, by using tools like TensorFlow Lite to decrease the size of the deep learning models. The goal of their system is to detect attacks on IoT devices using Convolutional Neural Networks (CNNs) and employs techniques such as pruning, weight clustering and post training quantization to lessen the model size and memory requirements without compromising much on accuracy.

System was tested on multiple low-power devices, including Arduino boards, ESP32 and Raspberry Pi and used a dataset designed to mitigate different IoT attack scenarios. The optimized model achieved high accuracy up to 99.74% on Raspberry Pi. While using very little power, ESP32 model achieved an accuracy of 97.21%. The system was even quick enough to make predictions in less than 1 microsecond on high-resource device like Raspberry Pi. However, some limitations are also discussed while deploying these models on extremely low-powered devices, such as Arduino Uno where RAM made it impractical to deploy even the optimized models. Researchers suggest that future work should be focused on further compressing the models or implementing custom hybrid models that could work in such constrained environments. This research directly aligns with the goal of this study of implementing lightweight anomaly detection systems for low-power IoT devices.
Antonini et al. (2023) proposed a novel, flexible anomaly detection system using TinyML for development in tough industrial environment, such as submersible pumps used in wastewater management plants. Their system uses TinyMLOps methods to handle the complexities of limited communication, accessibility and energy while guaranteeing reliable anomaly detection on low-cost microcontrollers like ESP32. Their system uses Isolation Random Forest algorithm, which is an unsupervised learning method, that can recognize anomalies without getting train on labeled data. It permits the system to detect errors in the device without relying on cloud support, which can be useful in harsh or remote environments where cloud connections are unreliable. To ensure data security, system logs detect anomalies using blockchain technology, which ensures data can't be changed. The system was evaluated in a simulated industrial environment and performed well, with models trained in 1.2 to 6.4 seconds and interference completed in less than 16 milliseconds using just 80kb of memory. While the system is effective but it still has some limitations when adapting to latest, unseen conditions without manual updates. To improve model adaptability in different surroundings, authors suggested to analyze dynamic on-device retraining and federated learning approach. They also suggested to integrate complex deep learning models to detect more delicate or intricate patterns in the data. This research is relevant to the current research as it depicts how lightweight TinyML models can be deployed in real-time for anomaly detection on low-powered edge devices even in harsh and resource constrained surroundings.

In conclusion, the literature points out the increasing demand for lightweight, efficient anomaly detection techniques in resource-constrained IoT environments. Although traditional machine learning techniques offer computational simplicity, they are not able to handle challenging anomaly patterns, whereas deep learning models give better accuracy at the expense of higher resource usage. The research shows that how compressed models such as hybrid methods and TinyML use less resources and detect anomalies accurately.

3 Research Methodology

Detection of unusual behavior and activities is important so that system works smoothly and efficiently. It is crucial in scenarios where quick decisions are required like monitoring devices that operate in real-time. This research involves the development of custom framework for solving the research problem that would help in anomaly detection in small and affordable devices like Raspberry Pi, which are widely used in Internet of Things (IoT) applications because of their portability, affordability, and computational capabilities. The primary challenge lies in detecting these unusual patterns effectively while remaining within the device's constraints, such as processing power and low memory. By identifying any unusual behavior in these components, potential problems can be recognized like software glitches, hardware breakdowns and hacking attempts by any unauthorized party.

3.1 Phases of our system

In this research, phases of our system involve data collection, pre-processing, training of model on OFF-Device (Centralized Environment), deploying compressed machine learning model for real-time interface on edge device such as Raspberry Pi on On-Device and evaluation. Figure 1 shows phases of our system.

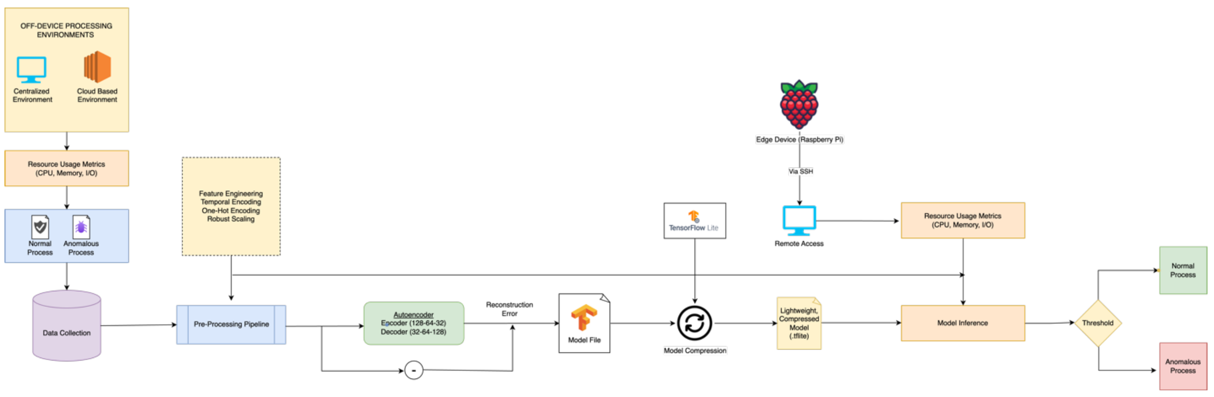

Figure 1: Phases of our System

3.1.1 Data Collection

Data collection is an essential part of this research as the relevance and quality of the data directly aligns with the accuracy of anomaly detection. Since the available datasets are not sufficient for process-level monitoring, especially for resource-constraint devices such as Raspberry Pi. This work is aimed at filling this gap by creating a synthetic dataset. A Synthetic dataset is a collection of artificially generated dataset that mimic the characteristics of real-world data. It helps researchers to have more control over their data generation process, enabling them to insert specific features, anomalies essential for training and testing the model (Kar, et al. 2019), (Paulin, et al. 2023). During the literature review, existing datasets like ToN-IoT (Inwa, etl a. 2024), BoT-IoT, and DS2OS (Huc, et al. 2021) were examined. Although all these datasets are useful for some certain applications, but they lack process-level data and fail to fulfill the specific requirement of our research. During literature review, we analyzed public datasets and found their key limitations:

- **Limitation of Process-Level Matrices:**

Majority of the public datasets concentrate on system-wide metrices, network traffic, application-level matrices but they do not have much data available for process-level monitoring. For example, ToN-IoT and BoT-IoT mainly concentrate on network traffic and lack on process-level monitoring. DS2OS gives general data of IoT but does not provide granular resource consumption of individual process. Anomaly detection at the process level needs information like detail on memory, CPU and other resource usage which above mentioned datasets do not provide.

- **Lack of Data at Real-Time Resource Usage**

Many existing datasets depend on simulated and historical data, which do not show real-time actions at process level. For instance, the work of (Skaperas, et al. 2024) focused on simulation of resource usage on cloud platforms but lacks to address real-time process level data that makes their approach less suitable for dynamic anomaly detection.

- **Inconsistency with Edge Devices**

Public datasets frequently ignored the constraints of resource-limited devices like Raspberry Pi. These devices work under limited CPU capabilities and memories and for real-time processing, lightweight datasets are required. For instance, the study of (Chen, et al. 2023) discussed the confrontation of the edge devices but existing datasets do not meet these certain requirements.

3.1.1.2 Creation of Dataset

Two scripts have been developed to overcome the above limitations:

Data Collector Script: It gathers the usage metrics of resources like CPU and memory, command line parameters, and network connections at every 60-second interval and stores the data in a CSV file.

Anomaly Generator Script: This anomaly generator script was design to simulate real-world spikes in CPU and memory usage, duplicating anomalous conditions that are crucial for training and evaluating the models. The design was inspired from multiple open-source resources and studies, such as CPULoadGenerator (Carlucci, 2019), Stack Overflow discussions (Stack Overflow, 2016), the Stress Injector from PyPI (PyPI, 2024), insights from the Qxf2 Blog (Shetty, 2023). The methodology of this research aligns with (Chouliaras et al. 2019), who used intentional stress workloads to create synthetic anomalies for detecting anomalies in NoSQL systems. Their work revealed that stress induces CPU spikes could distinguish abnormal behavior from normal operations. For example, their experiment showed that specific stress workloads prevailed in large CPU usage increases, marking them as abnormal. Similarly, our script applies controlled stress to mimic realistic anomalous scenarios, making sure that dataset involves both normal and abnormal signals for effective model training and testing. By running these scripts concurrently, we collected systematic data that represents both normal and anomalous behaviors, resulting in comprehensive dataset for detecting anomalies.

3.1.1.3 Dataset Features

The dataset contains following features which have been collected by running both of the scripts:

Table 2: Features of Dataset

Feature	Description
Timestamp	It includes second, minute, hour, day, month, year.
PID	Unique Process Identifier
Process Name	Name of the process
Username	User under which the process is executing.
Memory Usage (%)	Memory resources percentage

	used by the process.
CPU Usage (%)	CPU resources percentage used by the process.
Command Line	Parameters passed at run-time.
Connections	Network connections made by the process

3.1.1.4 Tools and Techniques

Various tools and techniques were considered for collecting data. To track matrices like I/O operations, CPU and memory usage, monitoring tools such as Grafana, Datadog, and Prometheus were considered. However, these tools are complex to set up and are not suitable for lightweight research or resource-limited environments like edge devices. In the same way, command-line utilities such as ps, htop, and top were considered. These are useful for real-time monitoring but they are insufficient for automation and programmability. Python library psutil was selected for this research due to its minimal computational overhead, feature set and efficiency. Psuil provides crucial data about network connections, I/O statics, and CPU and memory usage, making it best choice for environments like Raspberry Pi, where resource efficiency is utmost important.

3.1.2 Exploratory Data Analysis (EDA)

The EDA section presents the brief analysis of the dataset to understand the characteristics and distribution of processes for anomaly detection.

Table 3: Dataset Overview

Metric	**Value**	**Description**
Dataset Shape	(22937, 14)	Represents the shape of the dataset as rows x columns.
Missing Values	0	No missing values in any feature, ensuring the dataset is clean.
Unique Processes	363	Total unique processes running on the system.
Unique Users	2	Number of distinct users.
Numerical Features	CPU usage, Memory Usage	Numeric features available for resource utilization analysis.
Categorical Features	Process Name, Username	Categorical features to identify specific users and processes.

Mainly, the focus was on CPU and memory usage due to its critical role in detecting anomalies in resource constraint environments such as edge devices. High CPU and memory usage usually points out towards malicious activities like malware infections or DDoS attacks. (Lindqvist, et al. 1999) analyzed how unauthorized processes may give rise to peculiar spikes, and (Eskandri, et al. 2018) explained that attackers leverage vulnerabilities for cryptocurrency mining and DDoS attacks, which can lead to extreme resource utilization. According to (Lu, et al. (2011), another cause of anomalies can be software bugs, like memory leaks or infinite loops. This is particularly critical in resource-constrained devices, such as Raspberry Pi, they have limited computing power and can easily be saturated (Yang, et al, 2024). Thus, monitoring CPU and memory usage provides a strong foundation for detecting anomalies efficiently.

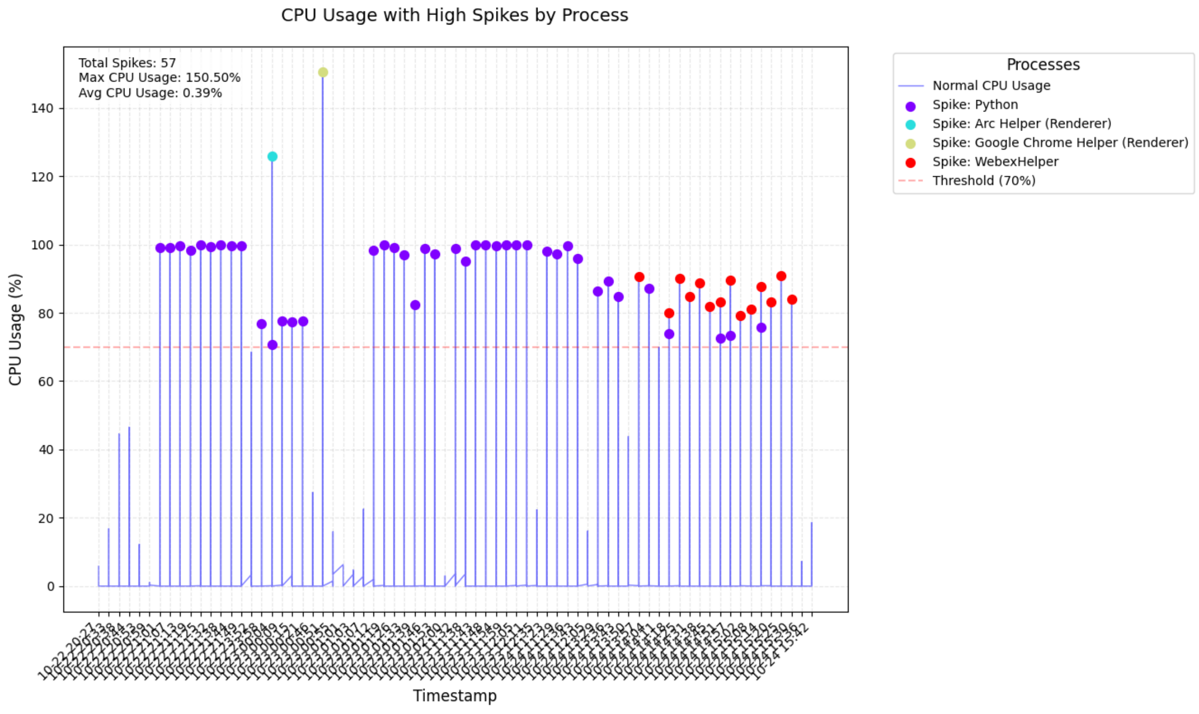

Figure 2: CPU usage with high spikes

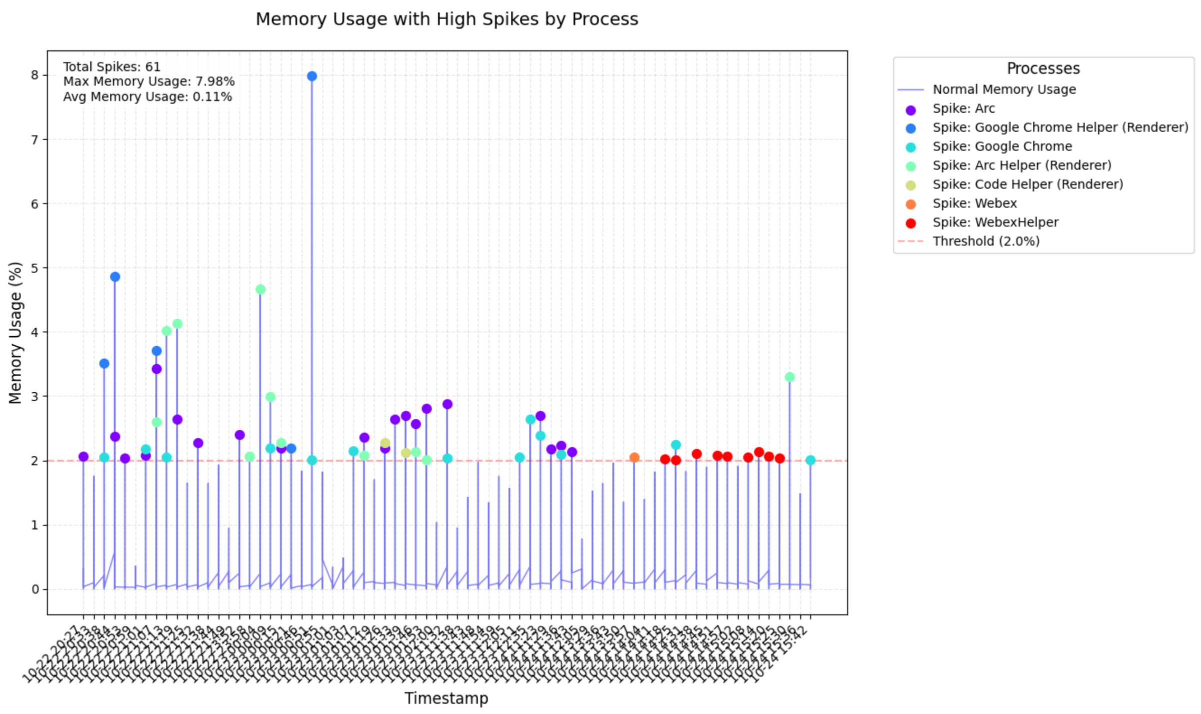

Figure 3: Memory usage with high spikes

3.1.3 Data Pre-processing

Data pre-processing is an essential step before model development as it scales, normalize the data for model which helps to train and evaluate the model. In this study, we have created the pre-processing pipeline for reusability instead of passing raw data through all functions and stored in pickle file. Following are the steps taken to process the data for detecting anomalies accurately and efficiently:

3.1.3.1 Data Cleaning

In data cleaning, several features were considered irrelevant in anomaly detection, and they were eliminated. Temporal features, such as, second, minute, day, month, year, timestamp was eliminated. This decision was taken as synthetic dataset is used along with random intervals for spikes, giving temporary patterns (like hourly or daily trends) irrelevant (Zhang, et al. (2019). Similarly, the process ID (PID), a unique identifier assigned to each process instance is also eliminated. It is helpful during data collection but lack predictive values for machine learning models, so we copied it into separate data frame for validating the model later. Command line feature was also removed because process name could be any and it can be changed so it holds no purpose.

3.1.3.2 Feature Transformation

Feature transformation is implemented on existing features to scale the data. Temporal features, such as Hour feature, was encoded using cyclic encoding method that includes sine and cosine transformation. where hour represents a value between 0 and 23. By applying these transformations, the cyclic relationship of time (for instance, 23:00 is close to 00:00) which helps to encode the hour into 0 and 1 and helps the model for better prediction. Cyclical encoding holds the periodic nature of time, helping the model to understand temporal patterns effectively. (Uber Engineering, 2019), (Ian, 2019). The Connection features, that display open network connection for each process were transformed into numerical value of active connections. This transformation is crucial as open network connections are frequently used in attacks, resulting in spikes in system resource usage like memory and CPU. Study of (Eskandri, 2018) and (Lindqvist, 1999) found out that open connections are key indicators of malicious activities like DDoS attacks. Additionally, a binary feature, root_user was inserted to make sure if the process was executed by root user. As processes executing at the root level have high privileges and are more prone to attacks, this feature helps in detecting anomalies that are connected to privilege escalation, improving the ability of the model to identify potential attacks.

3.1.3.3 Feature Encoding:

Feature encoding transforms categorical variables in a format that machine learning models can process. One-hot encoding helps in encoding categorical variable, Process_Name, generating binary features for each unique process name. It ensures that model treats each process as a unique category without suggesting any ordinal relationship. One-hot encoding is suitable for categorical data with low to medium number of unique values, as it maintains interoperability and prevents introducing bias.

3.1.3.4 Scaling Features:

Feature scaling adjusts numerical features to make sure they are on a similar scale, preserving features with larger values from overpowering the model's learning process. In this research, Robust Scaler scaled numerical features such as Memory_Usage(%), CPU_Usage(%) and Connections. To handle outliers, Robust Scaler is an effective choice as they are ideal for detecting anomalies in extreme values like CPU and Memory spikes. This method improves the performance of the model, especially in neural network architectures or distance-based metrices (Chouliaras, et al. 2019), (Zhang, et al. (2019).

3.1.4 Model Development

This section describes the development and training of autoencoder models for detecting anomalies in resource usage data. Initially, a basic baseline autoencoder was designed to understand data reconstruction and then progressed to more advance model to identify high-dimensional complexities. Additionally, fundamentals of autoencoder, their applications in anomaly detection, and the strategies employed for designing and training the model are discussed in this section.

3.1.4.1 Autoencoders

Autoencoders are neural networks used in unsupervised learning. It compresses the data into small, compact and meaningful representation via encoder and then recreates the original data with a decoder. The reconstruction error, the difference between the original and output data, helps to identify anomalies. Normal data has low construction errors while anomalies data have high errors. Autoencoders are popular in anomaly detection due to the fact that they can easily handle high-dimensional and mixed-type data, making them suitable for many applications. Unlike other datasets, they do not rely on labeled datasets and perform incredibly well with imbalanced datasets (Zong, et al. 2018). Their ability to operate efficiently on devices having limited resources make them suitable for real-world applications. Moreover, techniques like L2 regularization and dropout make them reliable by managing noisy data efficiently, making autoencoder a best choice to handling complex anomaly detection tasks (Chen, et al. 2020).

3.1.4.2 Baseline Model

The baseline model utilized a simple autoencoder architecture to test its basic ability to reconstruct data and detect anomalies. As (Song, et al. 2019) show in their research, that baseline autoencoder is efficient for simple anomaly detection tasks. Similarly, (Zong, et al. 2018) recommend starting with simple architectures to establish a foundation for further developments. Figure 2 shows baseline autoencoder architecture. It has the following structure:

1. **Input Layer:** Input layer size is same as the input feature.
2. **Hidden Layer:** It compresses information while taking the same size as the input data. It uses the linear activation function.
3. **Output Layer:** It reconstructs the input by using a linear activation function.

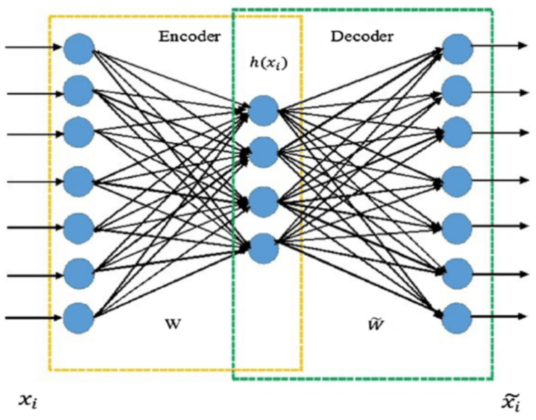

Figure 4: Baseline Model Architecture

The model was trained using Mean Squared Error (MSE) as the primary loss function, calculated using equation 2.

$$\mathrm{MSE} = \boxed{\frac{1}{n}}\ \boxed{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2} \qquad (1)$$

Where $\boxed{x_i}$ represents the mean value and $\boxed{\hat{x}_i}$ represents the constructive values. Mean Absolute Error (MAE) is also tackled as a second metric and is calculated in equation 3.

$$\mathrm{MAE} = \boxed{\frac{1}{n}}\ \boxed{\sum_{i=1}^{n}|x_i - \hat{x}_i|} \qquad (2)$$

The baseline model was trained on 20 epochs with a batch size of 256 and a validation split of 20%. Early stopping was used to ensure training stopped if the validation loss did not improve for 10 consecutive epochs. A learning rate scheduler reduced the learning rate by half if performance plateaued. While the model gave basic insights, its inability to manage non-linear patterns led to the development of an advanced architecture.

3.1.4.3 Advance Model

The advanced model sophisticatedly picked up more complex data patterns and greatly enhanced anomaly detection capabilities (Wang, et al. 2021) and (Zhang, et al. 2023) showed that advanced architectures better capture non-linear patterns. (Chen, et al. 2020) emphasized how dropout and L2 penalties are regularization techniques that make the model more robust. Figure 3 shows the pictorial representation of the advanced autoencoder architecture highlighting the encoder bottleneck decoder structure.

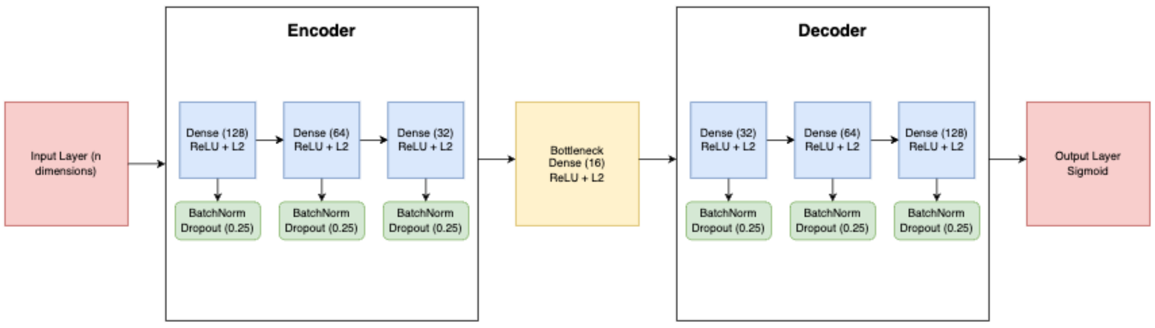

Figure 5: Advance Autoencoder Architecture

To overcome the weaknesses of the baseline model, an advanced autoencoder has been used with architectural improvements.

- **Encoder**

The encoder involved multiple layers with fewer neurons at each step: 128 → 64 → 32 → 16. It compresses the input step by step. The ReLU activation functions were used to capture complex relationships in the data. Dropout with 25% and batch normalization were used after each layer to improve generalization and stabilize training.

- **Latent Space (Bottleneck)**

The latent space was a dense layer with 16 neurons that formed a compact summary of the data. This bottleneck layer acted effectively in filtering out noise and redundancy while retaining the important structure of the input.

- **Decoder**

The decoder was a mirrored version of the encoder, where each layer had an equal size: 16 → 32 → 64 → 128, to reconstruct the input. Sigmoid activation in the output layer ensures that reconstructed values are in the normalized range of the input features.

3.1.5 Model Compression

Model compression plays a crucial role in the deployment of machine learning models in resource constraint environments like edge devices. It lessens the need for computational power and storage while preserving accuracy. To compress autoencoder model, TensorFlow Lite was implemented in this study. It enables efficient, lightweight and scalable real-time anomaly detection on devices including Raspberry Pi. TensorFlow Lite, created by Google, is a framework designed to efficiently deploy machine learning models on mobile, IoT and embedded devices. It integrates ease with effective model compression, increase inference speed, making it optimal for resource efficient deployment in real-time applications (Howard, et al. 2019). By transforming model into compact format, TensorFlow Lite reduces the size of the model, making it convenient for edge devices with limited resources. It also increases the speed of the inference by using optimization methods such as kernel optimization and operator fusion, which are significant for real-time anomaly detection (Zhou, et al. 2021). Moreover, due to post-training quantization support of TensorFlow Lite, 32-bit floating-point weights are compacted into 8-bit integers. This process reduces the size of the model and power consumption (Jacob, et al. 2018). With the help of cross-platform development capabilities of TensorFlow Lite, deploying anomaly detection models on various edge devices have become easy (Howard, et al, 2019). Its API makes the workflow easy by allowing flawless conversation of trained TensorFlow/Keras models into deployable formats. In contrast to other compression methods, TensorFlow Lite provides a comprehensive solution. For instance, pruning lessens the size of the model by eliminating unnecessary connections but to maintain the performance, it needs retraining (Han, et al. 2015). Knowledge distillation delivers knowledge from large model to small model but adds extra complexity during training (Hinton, et al. 2015). In contrast, TensorFlow Lite integrates inference optimization, size reduction and quantization into single, effective framework, making it extensive choice for real-world deployment.

3.1.6 Model Inference

In this section, a lightweight version of our model, TensorFlow Lite (TFLite) is used for anomaly detection as well as real-time monitoring of process-level resource usage. This model is lightweight and efficient and is designed particularly for devices with low processing power. The process is initiated by gathering real-time data, which includes memory consumption, CPU usage, network activity, and the names of active processes that are collected every 5 seconds. This frequent data collection is carried out to detect anomalies with putting stress on the system. Data goes through multiple preparation steps before analysis so that it meets model's requirement. This includes irrelevant detail removal, scaling numbers to ensure consistency, rearranging the data layout according to what the model expects, and encoding categorical variables. Model tries to reconstruct input data according to the patterns that it learned during the training process. If reconstruction error is greater than a certain threshold, system flags it as an anomaly. This inference helping to detect real-time anomalies on resource usage of processes without putting overhead on limited reources of the system.

4 Evaluation

In this section, we evaluated the performance of our baseline and advanced autoencoder models to check the effective identification of anomalies in resource usage processes. We split the dataset into 80% training (64%) and 20% testing (20%). During training, 20% of the training data was used for validation to ensure proper evaluation on unseen data. The evaluation took place by examining the training history of each model, including visualization of loss and mean absolute error (MAE) across epochs. These visualizations gave insights on how well each model learned and generalized over time. To further test the trained models, predictions against the original data were compared for reconstruction errors. The results were visualized through histograms of reconstruction errors and scatter plots of CPU vs. memory usage, providing insights to understand the distribution of anomalies and the relationship between resource metrics. In addition, confusion matrices have been used to measure the classification performance of the models. A confusion matrix breaks down the predictions into true positives (properly detected anomalies), true negatives (properly identified normal processes), false positives (normal processes misclassified as anomalies), and false negatives (anomalies missed by the model). Important metrics like accuracy, precision, recall, and F1-score were computed to determine the quantitative performance of the model. Accuracy measures the overall preciseness of the model, precision is used to measure the significance of detected anomalies, recall is used to measure the sensitivity over anomaly detection, and the F1-score measures the balance in between precision and recall. Table 3 shows devices used for model testing and evaluation.

Table 4: Devices Used for Model Testing and Evaluation

Devices	Purpose	Specifications
MacBook Pro (M1)	Model training and dataset testing.	**Chip**: Apple M1 **Memory**: 8 GB **macOS**: Sequoia 15.1.1
Raspberry Pi Zero 2W	Real-time anomaly detection and testing.	**CPU**: 1 GHz quad-core ARM Cortex-A53 **Memory**: 512 MB SDRAM **OS**: Raspberry Pi OS

4.1. Model Architecture Evaluation

We evaluated the baseline and advanced models by analyzing their training and validation performance, focusing on loss and Mean Absolute Error (MAE) metrics, as well as their ability to distinguish between normal and anomalous behaviors. For the baseline model, we observed a sharp decrease in both training and validation loss during the initial epochs, which stabilized around epoch 5. Figure 4 visually illustrates the training loss and MAE for the baseline model. The validation MAE closely matched the training MAE, indicating minimal overfitting. To understand the model's ability to detect anomalies, we plotted histograms of reconstruction errors, which showed a clear separation between normal and anomalous data points. Figure 5 shows the Reconstruction Error Distribution and CPU vs. Memory Usage for the baseline model. However, we noted that the threshold margin was relatively narrow, suggesting limitations in capturing more complex patterns. Additionally, we used scatter plots of CPU versus memory usage to visualize the anomalies, where some overlap with normal data points were observed, indicating room for improvement in the model's generalization. In contrast, the advanced model exhibited smoother and more gradual convergence compared to the baseline, with both training and validation loss consistently decreasing and stabilizing around epoch 20. Figure 6 depicts the Model Loss and MAE for the advanced model. The advanced model achieved substantially lower values of MAE and loss, which indicated better learning efficiency. The error histogram also showed a sharper separation between normal and anomalous data with a wider threshold margin, which indicated better anomaly detection capabilities. Also, in CPU vs. memory usage, the advanced model was more accurate in anomaly detection, where anomalous points were clearly, signifying improved generalization and robustness. Figure 7 shows the advanced model Reconstructions Error Distribution with more details on how anomalies are better separated. The advanced model outperformed the baseline in training convergence, error separation, and anomaly accuracy detection. Addition of extra hidden layers, dropout, and batch helped this model perform better than the baseline. Normalization in the advanced architecture improved the ability of the model to learn complex patterns, making it more effective for real-world anomaly detection scenarios. This enhanced performance highlights the advanced model's suitability for resource usage anomaly detection tasks, demonstrating its potential for practical applications.

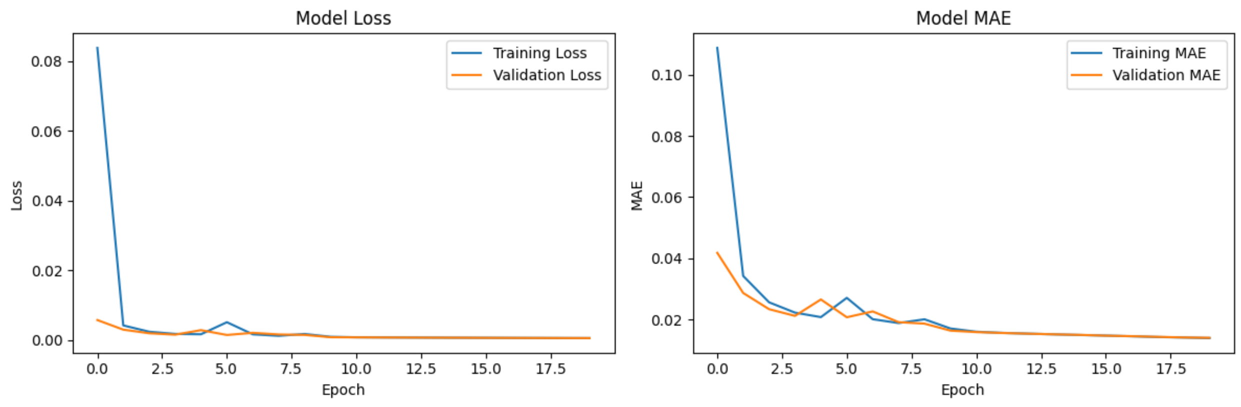

Figure 6: Training Loss and MAE for baseline model

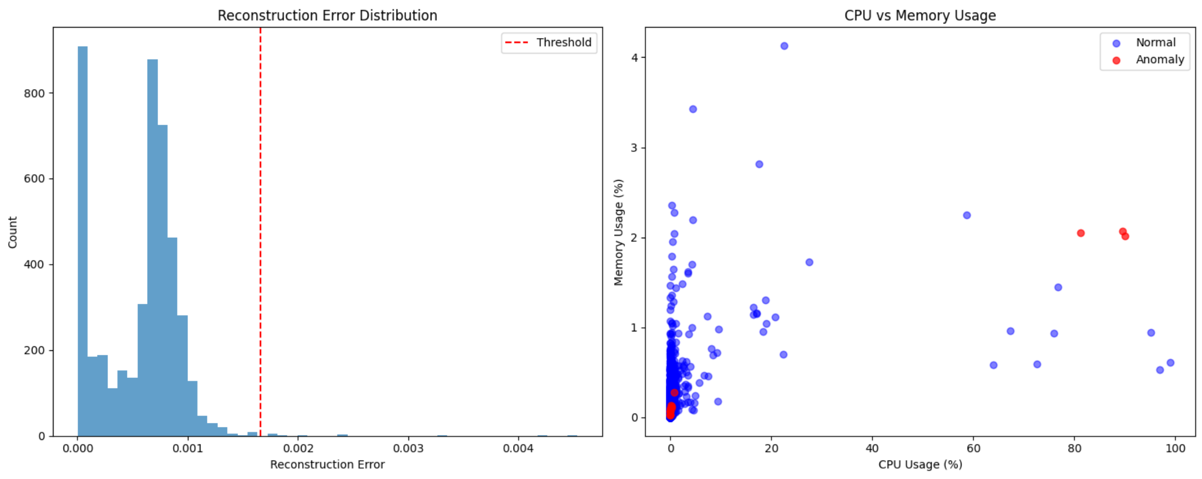

Figure 7: Reconstruction Error Distribution and CPU vs Memory Usage for Baseline Model

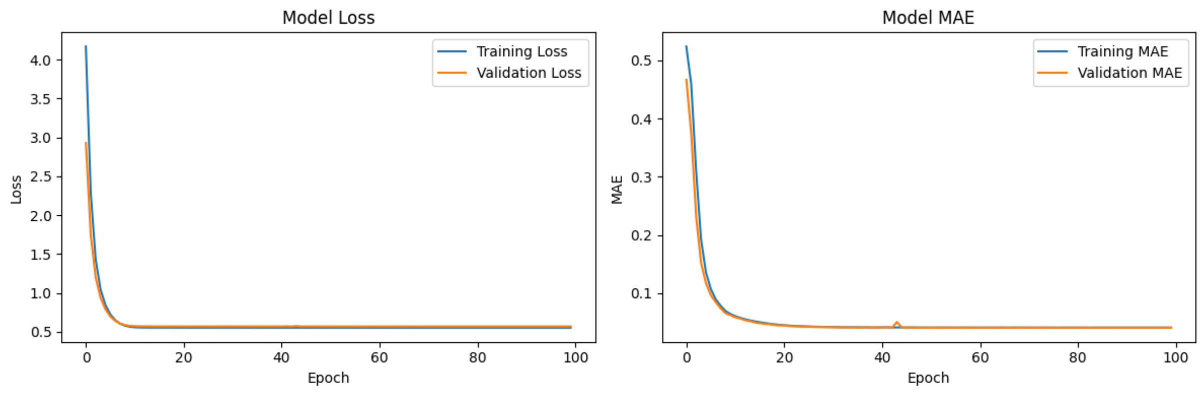

Figure 8: Model Loss and MAE for Advanced Model.

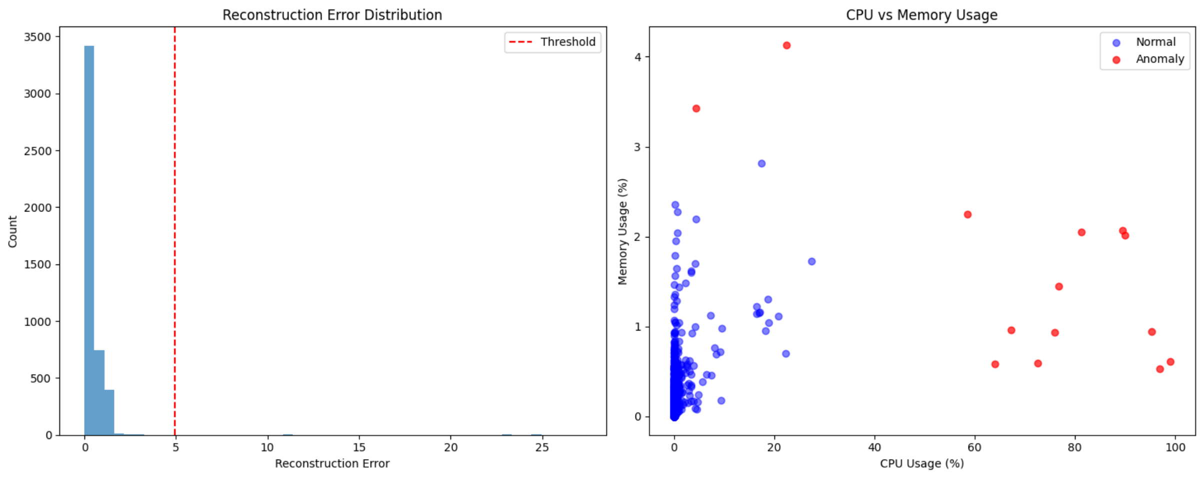

Figure 9: Reconstruction Error Distribution for the advanced model

4.2. Active Model vs Compressed Model

Model performance was compared in actual and compressed models on the Raspberry Pi dataset. Table 4 displays the performance metrics including, accuracy, precision, recall, F1 score, and confusion matrix components, both models having same file size. Figure 8 displays the confusion matrices of actual and compressed models of their classification results.

Table 5: Actual and compressed model performance comparison on Raspberry Pi dataset.

Model Type	**Model File Size**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**True Negatives (TN)**	**False Positives (FP)**	**False Negatives (FN)**	**True Positives (TP)**
Actual Model	1,517,920 bytes (1.5 MB)	0.9700	1.0000	0.3022	0.4641	3091	0	97	42
Compressed (Lite)	1,517,920	0.9700	1.0000	0.3022	0.4641	3091	0	97	42

Model	bytes (1.5 MB)								

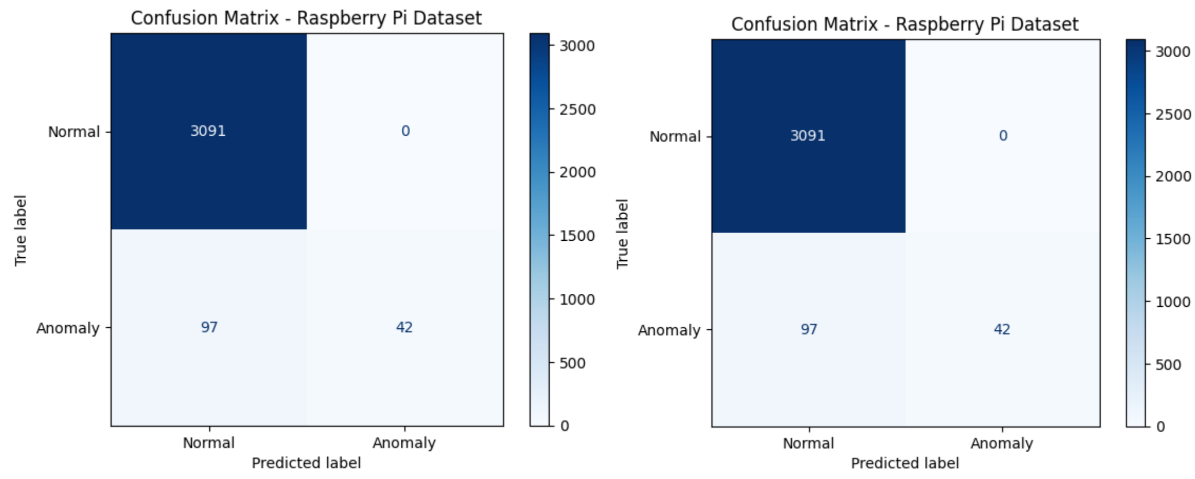

Figure 10: Confusion matrices for the actual and compressed models on Raspberry Pi dataset.

4.3. Model Inference Evaluation

The Raspberry Pi Zero 2W was used as the deployment device to assess the real-time anomaly detection capability of the model. This was done by establishing an SSH connection to the Raspberry Pi and accessing the project directory, which contained both the inference and anomaly generator scripts. The model inference script was executed in parallel with the anomaly generator script. The anomaly generator script was simulated to make the system show anomalous behaviors by producing processes consuming much CPU and memory resources. Meanwhile, the inference script monitored the system real-time, analyzing resource usage and classifying processes as normal or anomalous based on the pre-trained model.

5 Discussion, Conclusion & Future Work

In this study, the process had to overcome several technical challenges to make it robust and practical. While data collection process, the major challenge was to capture CPU and memory usage accurately. At first, the use of psutil library with an interval=0 failed to capture high-frequency spikes. From the documentation of the library, it was realized that setting interval=1 gives accurate average usage over one second, which matches the data collection frequency of 60 seconds. To counter the blocking nature of this configuration, the data logging script was redesigned asynchronously, thus ensuring efficiency and avoiding delays in data collection. Designing the anomaly generator script was also a challenge as early iterations resulted in crashes and excessive memory usage that were unacceptable for resource-constrained environments. The simulating of CPU and memory spikes was refined to ensure that the script would run smoothly without overloading the system, hence its seamless use on edge devices to provide realistic conditions for model evaluation. A limitation between precision and recall was observed due to the sigma anomaly threshold which flagged anomalies deviating above 95%. The accuracy was better but it was missing some anomalies because we labelled the data above 70% usage of spikes as anomalies for validating the model but adaptative threshold could improve the recall and precision.

Our research addressed the problem of anomaly detection in resource usage processes on resource-constrained edge devices. Motivated by the question, how effectively can a compressed, lightweight machine learning model detect anomalous activities on resource usage of processes using edge devices? Our research has successfully achieved its aim. A robust anomaly detection system was developed using autoencoder architectures that were trained on resource usage data. The advance model, with improvements such as additional hidden layers, dropout regularization, and batch normalization performed much better in terms of learning efficiency and anomaly detection over the baseline model. To overcome resource constraints, the trained model was compressed with TensorFlow Lite in order to get a lightweight version with retained performance, drastically decreasing computational and memory requirements such that deployment would be feasible in low-resource environments. The compressed model was deployed onto a Raspberry Pi Zero 2W, where in real-time, it provided anomaly detection. It made accurate identification of anomalous processes in live data streams, affirming its practical applicability for edge computing. With this deployment, the entire system was shown to balance efficiency and performance in resource-constrained environments. Our work takes a significant step in closing the gap between high performance machine learning and edge deployments.

Future work will continue to improve the robustness and usability of the anomaly detection system by overcoming current limitations and exploring new directions. Gathering more diverse workloads and datasets from different IoT devices and environments will be a priority. Leveraging platforms like OpenWrt OS, which offers flexibility in managing networked devices, could enable broader data collection and testing under varied conditions. A real-time alerting system is also going to be the focus of development. It will notify users or administrators about detected anomalies. This approach enhances the practicality by enabling swift responses to potential threats. Incorporating adaptive threshold mechanisms could further increase the detection criteria to adjust dynamically based on system behavior and workload patterns, optimizing the balance between precision and recall. Moreover, establishing a user-friendly interface for managing alerts, monitoring resource usage and visualizing results would make the system more accessible to non-technical users in industrial and smart home settings. Advanced model architectures, such as graph neural networks and transformer-based models could also be explored for enhancing the system's ability to detect complex patterns in resource usage data effectively. By making these developments, the system could evolve into an efficient and scalable solution for real-time anomaly detection across diverse IoT ecosystems.

Acknowledgement

I am profoundly grateful to my supervisor Mr. Vikas Tomer, for his exceptional guidance and unwavering support throughout my thesis journey. His continuous encouragement and constructive feedback not only helped me complete my thesis but also enhanced my professional expertise and skills. It has been a great experience for me to complete my thesis under his supervision. I extend my heartfelt appreciation to my elder brother, Shavaiz Shamas, my family, and friends, whose constant support, understanding, and motivation were crucial in helping me navigate the challenges of this academic journey. Lastly, I would like to express my sincere thanks to the entire faculty of NCI for their guidance, mentorship, and support thorough out my master’s program.

References

Aldossari, Mobark & Sidorova, Anna. (2018). Consumer Acceptance of Internet of Things (IoT): Smart Home Context. Journal of Computer Information Systems. 60. 1-11. 10.1080/08874417.2018.1543000.

Alyasiri, H., Clark, J.A., Malik, A. and de Fréin, R., 2021, July. Grammatical evolution for detecting cyberattacks in Internet of Things environments. In 2021 International Conference on Computer Communications and Networks (ICCCN) (pp. 1-6). IEEE.

Al-amri, R. et al. (2021) "A review of machine learning and deep learning techniques for anomaly detection in IoT data," Applied sciences (Basel, Switzerland), 11(12), p. 5320. Available at: https://doi.org/10.3390/app11125320.

Antonakakis, M. (no date) Understanding the Mirai Botnet, Usenix.org. Available at: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf (Accessed: August 1, 2024).

Antonini, M. et al. (2023) "An adaptable and unsupervised TinyML anomaly detection system for extreme industrial environments," Sensors (Basel, Switzerland), 23(4), p. 2344. Available at: https://doi.org/10.3390/s23042344.

Antonioli, D., Tippenhauer, N.O., Rasmussen, K. and Payer, M., 2022, May. Blurtooth: Exploiting cross-transport key derivation in bluetooth classic and bluetooth low energy. In Proceedings of the 2022 ACM on Asia conference on computer and communications security (pp. 196-207).

Breitenbacher, D., Homoliak, I., Aung, Y.L., Tippenhauer, N.O. and Elovici, Y., 2019, July. HADES-IoT: A practical host-based anomaly detection system for IoT devices. In Proceedings of the 2019 ACM Asia conference on computer and communications security (pp. 479-484).

Carlucci, G. (no date) CPULoadGenerator: CPU Load Generator allows you to generate a fixed configurable CPU load for a finite time by means of PID regulator.

Chatterjee, A. and Ahmed, B.S. (2022) "IoT anomaly detection methods and applications: A survey," Internet of Things, 19(100568), p. 100568. Available at: https://doi.org/10.1016/j.iot.2022.100568.

Chen, J., Wang, Z., & Wu, T. (2020). Regularization strategies for deep anomaly detection. IEEE Transactions on Neural Networks and Learning Systems. https://ieeexplore.ieee.org/document/8963456.

Chouliaras, S. and Sotiriadis, S. (2020) "Real-time anomaly detection of NoSQL systems based on resource usage monitoring," IEEE transactions on industrial informatics, 16(9), pp. 6042–6049. Available at: https://doi.org/10.1109/tii.2019.2958606.

Dastjerdi, A.V. and Buyya, R., 2016. Fog computing: Helping the Internet of Things realize its potential. Computer, 49(8), pp.112-116.

Eskandari, S. et al. (2018) "A First Look at Browser-Based Cryptojacking," in 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, pp. 58–66.

Fanariotis, A. et al. (2023) "Power efficient Machine Learning models deployment on edge IoT devices," Sensors (Basel, Switzerland), 23(3), p. 1595. Available at: https://doi.org/10.3390/s23031595.

Gharavi, H., Granjal, J. and Monteiro, E., 2024. Post-quantum blockchain security for the Internet of Things: Survey and research directions. IEEE Communications Surveys & Tutorials.

Gudala, et al., (2019) "Leveraging Artificial Intelligence for Enhanced Threat Detection, Response, and Anomaly Identification in Resource-Constrained IoT Networks", Distributed Learning and Broad Applications in Scientific Research, 5, pp. 23–54. Available at: https://dlabi.org/index.php/journal/article/view/4 (Accessed: 7 December 2024).

Haji, S. and Ameen, S., 2021. Attack and anomaly detection in IoT networks using machine learning techniques: A review. Asian Journal of Research in Computer Science, [online] 9(2), pp.30-46. Available at: https://doi.org/10.9734/ajrcos/2021/v9i230218 [Accessed 4 Dec. 2024].

Haowen, X. et al. (2018) "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in Web applications," arXiv [cs.LG]. Available at: http://arxiv.org/abs/1802.03903.

Huc, A., Salej, J. and Trebar, M. (2021) "Analysis of machine learning algorithms for anomaly detection on edge devices," Sensors (Basel, Switzerland), 21(14), p. 4946. Available at: https://doi.org/10.3390/s21144946.

Idrissi, I., Moussaoui, O. and Azizi, M. (2022) "A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for IoT," International Journal of Computing and Digital Systems, 11(1), pp. 209–216. Available at: https://doi.org/10.12785/ijcds/110117.

Inuwa, M.M. and Das, R. (2024) "A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks," Internet of Things, 26(101162), p. 101162. Available at: https://doi.org/10.1016/j.iot.2024.101162.

Jove, E., Aveleira-Mata, J., Alaiz-Moretón, H., Casteleiro-Roca, J.L., Marcos del Blanco, D.Y., Zayas-Gato, F., Quintián, H. and Calvo-Rolle, J.L., 2022. Intelligent one-class classifiers for the development of an intrusion detection system: the mqtt case study. Electronics, 11(3), p.422.

Kar, A., Prakash, A., Liu, M.Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A.and Fidler, S., 2019. Meta-sim: Learning to generate synthetic datasets. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4551-4560).

Koroniotis, N., Moustafa, N., Sitnikova, E. and Turnbull, B., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Generation Computer Systems, 100, pp.779-796.

Landauer, M. et al. (2023) "Deep learning for anomaly detection in log data: A survey," Machine learning with applications, 12(100470), p. 100470. Available at: https://doi.org/10.1016/j.mlwa.2023.100470.

Li, Y., Mandalari, A.M. and Straw, I., 2023. Who let the smart toaster hack the house? An investigation into the security vulnerabilities of consumer IoT devices. arXiv preprint arXiv:2306.09017.

Lindqvist, U. and Porras, P.A. (1999) "Detecting computer and network misuse through the production-based expert system toolset (P-BEST)," Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344), pp. 146–161. Available at: https://doi.org/10.1109/SECPRI.1999.766911.

Lu, Shan & Park, Soyeon & Seo, Eunsoo & Zhou, Yuanyuan. (2008). Learning from mistakes: A comprehensive study on real world concurrency bug characteristics. SIGARCH Computer Architecture News. 36. 329-339. 10.1145/1346281.1346323.

Mota, A., Serôdio, C. and Valente, A., 2024. Matter Protocol Integration Using Espressif's Solutions to Achieve Smart Home Interoperability. Electronics, 13(11), p.2217.

Nkuba, C.K. et al. (2023) "ZMAD: Lightweight model-based anomaly detection for the structured Z-wave protocol," IEEE access: practical innovations, open solutions, 11, pp. 60562–60577. Available at: https://doi.org/10.1109/access.2023.3285476.

Paulin, G. and Ivasic-Kos, M., 2023. Review and analysis of synthetic dataset generation methods and techniques for application in computer vision. Artificial intelligence review, 56(9), pp.9221-9265.

Premsankar, G., Di Francesco, M. and Taleb, T. (2018) "Edge computing for the internet of things: A case study," IEEE internet of things journal, 5(2), pp. 1275–1284. doi: 10.1109/jiot.2018.2805263.

Raghavendra, C. and Sanjay, C. (2019) "Deep learning for anomaly detection: A survey," arXiv [cs.LG]. Available at: http://arxiv.org/abs/1901.03407.

Says, G., 2019. 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020. Gartner. Available online: https://www. gartner. com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io (accessed on 12 July 2021).

Schiller, E., Aidoo, A., Fuhrer, J., Stahl, J., Ziörjen, M. and Stiller, B., 2022. Landscape of IoT security. Computer Science Review, 44, p.100467.

Shetty, A. (2023) Generate CPU load using Python, Qxf2 BLOG. Qxf2 Services. Available at: https://qxf2.com/blog/generate-cpu-load/ (Accessed: December 11, 2024).

Siang, Y. Y. ., Ahamd, M. R. . and Zainal Abidin, M. S. (2021) "Anomaly Detection Based on Tiny Machine Learning: A Review", Open International Journal of Informatics, 9(Special Issue 2), pp. 67–78. doi: 10.11113/oiji2021.9nSpecial Issue 2.148.

Song, H., Rajasegarar, S., & Leckie, C. (2019). A survey of deep learning techniques for anomaly detection. arXiv preprint. https://arxiv.org/pdf/1901.03407.

Sotiris, S. et al. (2024) "A pragmatical approach to anomaly detection evaluation in edge cloud systems," arXiv [cs.NI]. Available at: http://www.intersys-lab.org/media/papers/2024/pragmatical-approach.pdf (Accessed: December 7, 2024).

Tekin, N. et al. (2023) "Energy consumption of on-device machine learning models for IoT intrusion detection," Internet of Things, 21(100670), p. 100670. Available at: https://doi.org/10.1016/j.iot.2022.100670.

Wang, X., Li, Y., & Zhang, Y. (2021). Autoencoders for anomaly detection in high-dimensional data. Applied Sciences. https://www.mdpi.com/2076-3417/11/12/5320#B19-applsci-11-05320.

Wang, Z. et al. (2022) "LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge," Computer networks, 203(108616), p. 108616. Available at: https://doi.org/10.1016/j.comnet.2021.108616.

Wu, J., Nan, Y., Kumar, V., Tian, D.J., Bianchi, A., Payer, M. and Xu, D., 2020. {BLESA}: Spoofing attacks against reconnections in bluetooth low energy. In 14th USENIX Workshop on Offensive Technologies (WOOT 20).

Yang, Min & Zhang, Jiajie. (2023). Data Anomaly Detection in the Internet of Things: A Review of Current Trends and Research Challenges. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.0140901.

Zhang, Q. et al. (2021) "Lightweight and accurate DNN-based anomaly detection at edge," IEEE transactions on parallel and distributed systems: a publication of the IEEE Computer Society, 33(11), pp. 1–1. Available at: https://doi.org/10.1109/tpds.2021.3137631.

Zhang, Y., Zhou, D., Chen, S., & Xu, J. (2019). Challenges in designing anomaly detection models for operational systems. arXiv preprint. https://arxiv.org/pdf/1906.03821.

Zhang, Y., Zhou, D., & Xu, J. (2023). Machine learning models for anomaly detection in high-dimensional data. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S2666827023000233.

Ziegler, T. (2023) Applications of AI on Resource-ConstrainedHardware with a focus on Anomaly Detection. Massachusetts Institute of Technology.

Zong, B., Song, Q., Huang, H., Swanson, E., Eide, D., Ding, Y., & Han, W. (2018). Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. International Conference on Learning Representations (ICLR). https://arxiv.org/pdf/1802.03903.