uni.systems
Research, Development & Innovation (RDI) Department
Data Science & Cyber Security

Intrusion Detection System in IoT
Deep Learning Approach

Nikitas Tsinnas
ECE NTUA Graduate - RDI Trainee

September - October
2023

Contents

1 Introduction

The goal of this writing is the documentation of the concepts, tools and technical procedures that I learned through my practical training in the Research, Development and Innovation (RDI) department of uni.systems. I was left free to choose which research topic I wanted to work on and was given appropriate guidance to understand the difference between research and production. More specifically, instead of only focusing in machine learning model design, there was an attempt to simulate how the research and production can be bridged through building an Experience Canvas, designing a solution architecture and encountering technical challenges.

1.1 Internet of Things

A Coca-Cola vending machine, in 1982 at Carnegie Mellon University, was the first connected-appliance to the ARPANET -the forerunner of the modern internet. This machine was able to report its current inventory and temperature of the newly loaded drinks through the network. After 9 years, a contemporary vision of the IoT was produced by a computer scientist called Mark Weiser. Since then, especially at current times, the term is used quite often. Today, when someone refers to "Internet of Things" he describes a set of devices with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

In the consumer market, the IoT technology is directly related to "smart home" products including devices and applicances such as lightbulbs, thermostats, home security systems, cameras, speakers and other home appliances and can be part of a common ecosystem in which the same or other devices connected to this ecosystem can control and monitor the rest of the devices. In addition to smart homes, the IoT frameworks include industrial infrastructures such as smart grids in the energy sector. The technologies that make today's IoT-enabled energy grid "smart" include wireless devices such as sensors, radio modules, gateways and routers. These devices provide the sophisticated connectivity and communications that empower consumers to make better energy usage decisions and they are able to do these by connecting themselves to the internet.

From home security appliances to industrial sensors, all those small devices are connected to the internet through their ecosystem's gateway and thus they are a part of the IoT industry.

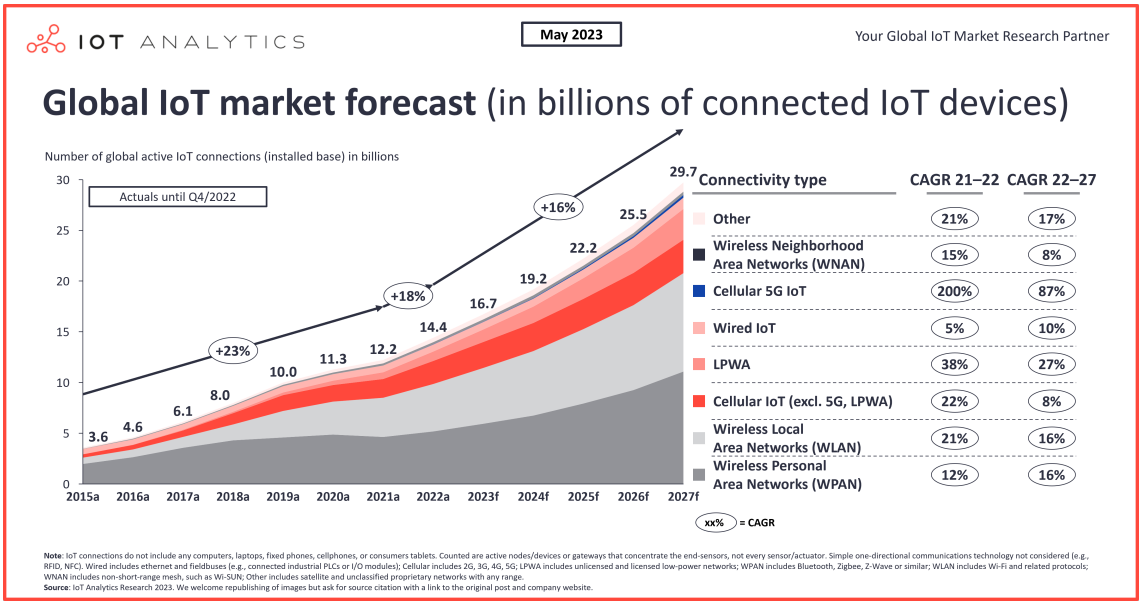

Figure 1: IoT Global Industry forecast

To summarize, IoT involves adding internet connectivity to a system of interrelated computing, monitor and control devices of mechanical and digital nature. Each "thing" has a unique identifier and the ability to automatically transfer data over a network.

1.2 Cyber Security in IoT

However, enabling devices to connect to the internet opens them up to serious vulnerabilities. Due to the unconventional manufacturing of IoT devices and the vast amount of data they handle, there is a constant threat of cyber attacks. The two factors that have led to an exposure of IoT vulnerabilities to cyber threats are: large-scale proliferation of IoT devices from within households all the way to critical infrastructures including smart electricity

grids and smart vehicles, and the heterogeneity in the communication protocols of IoT devices.

The numbers of tools and experimental platforms that are available to run the IoT device and network-level simulations and experiments, are ever-increasing. One such protocol, namely, the Message Queuing Telemetry Transport (MQTT), is an IoT connectivity protocol, which is a lightweight publish-subscribe protocol to facilitate the remote connectivity between IoT devices and centralised brokers or base stations. Due to its lightweight nature, that facilitates low bandwidth, low power and low storage on IoT devices, the MQTT protocol can be easily deployed on mobile applications as well. The MQTT protocol suffers from many vulnerabilities that can be exploited by the adversary. Some of the threats against the protocol include: device compromise, data privacy, Denial of Service (DoS) against MQTT service, and Man-inThe-Middle (MiTM) attacks against MQTT messages.

Several incidents where a common IoT device was used to infiltrate and attack the larger network have drawn attention to the need for IoT security. Many IoT devices are set up with the default factory settings which means that the administrator username and password are not changed by the average user who is not aware of the dangers. This is a vulnerability that Mirai malware took advantage. In this type of attack, infected devices continuously scan the internet for the IP address of IoT devices. Mirai then identifies vulnerable IoT devices using a table of more than 60 common factory default usernames and passwords, and logs into them to infect them with the Mirai malware. Infected devices will continue to function normally, except for occasional sluggishness, and an increased use of bandwidth in their local network.

Mirai malware was first used in 20 September 2016 in the DDoS on the Krebs on Security site which reached 620 Gbit/s. Ars Technica also reported a 1 Tbit/s attack on French web host OVH.On 21 October 2016, multiple major DDoS attacks in DNS services of DNS service provider Dyn occurred using Mirai malware installed on a large number of IoT devices, many of which were still using their default usernames and passwords. These attacks resulted in the inaccessibility of several high-profile websites, including GitHub, Twitter, Reddit, Netflix, Airbnb and many others. Therefore IoT security is crucial and it constitutes a continuous concern in a global scale, not only because of the danger of a potential private data leakage or un-

wanted external control of IoT devices, but also because of the vast number of connected nodes that can be manipulated from attackers to empower their malicious actions.

Utilizing artificial intelligence (AI) to promptly identify malicious attacks is essential and highly efficient. The realm of AI methods applicable to network intrusion detection, which involves the analysis of network traffic to spot malicious attacks, is quite extensive. A variety of AI techniques, including support vector machines (SVM), Bayesian networks, principal component analysis (PCA), and genetic algorithms (GA), have been widely employed to detect patterns of abnormal behavior within IoT networks. Nonetheless, deep learning-based intrusion detection for IoT networks remains a relatively unexplored area.

1.3 Deep Learning

Deep learning falls within the broader category of machine learning techniques which are part of the broader science of Artificial Intelligence. Data science involves analyzing data to discover valuable insights for businesses. It's a multidisciplinary approach that combines concepts from mathematics, statistics, artificial intelligence, and computer engineering to study large datasets.

Deep learning is a subset of machine learning algorithms that utilize multiple layers to incrementally extract higher-level characteristics from the original input data. For instance, in the context of image processing, the lower layers might identify basic features like edges, whereas the upper layers could recognize more complex human-relevant elements such as numbers, letters, or faces. Most modern deep learning models are based on multi-layered artificial neural networks such as convolutional neural networks and transformers.

The phrases "neural network" and "deep learning" are frequently interchanged, yet there exist subtle distinctions between them. Although both fall under the umbrella of machine learning, a neural network emulates the functioning of biological neurons in the human brain, whereas a deep learning network consists of multiple layers of neural networks. In the human brain, neurons are interconnected, enabling the reception, processing, and transmission of information. Analogously, artificial neural networks operate through an organized structure consisting of an input layer, one or more hidden lay-

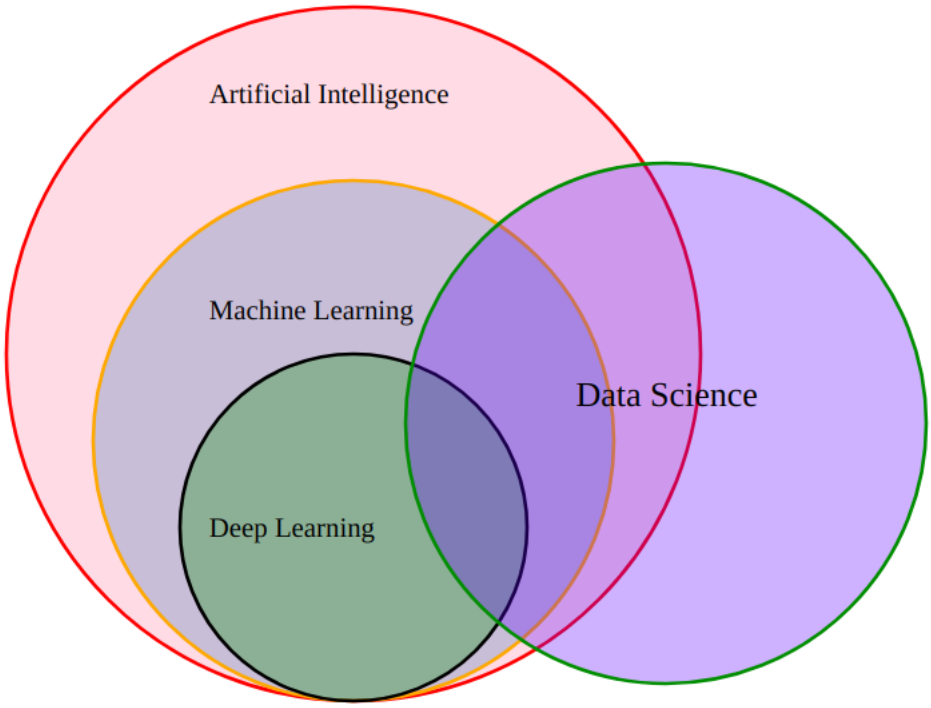

Figure 2: AI Venn Diagram

ers, and an output layer. Thus, a basic neural network comprises these three essential components: input, hidden, and output layers. Conversely, deep learning is a more sophisticated paradigm. It entails the utilization of multiple hidden layers, each composed of interconnected neurons. Deep neural networks exhibit exceptional proficiency in handling extensive and diverse datasets encompassing various types, such as images, text, audio, or time series data. These networks undergo a process of iterative learning, leveraging training data to enhance their capacity for recognizing intricate patterns within real-world data, progressively refining their predictive accuracy over time.

Neural networks can be trained to "learn" and recognize hidden relationships, patterns, and trends in data. Being able to recognise items in pictures is a sophisticated task that even humans cannot documentate the process of their thoughts in order to implement this exact logic to a computational

machine. Neural networks is mimicking the human's brain structure in the machine language using mathematical logic concepts. The way this works is this: In the first step, the initial layer takes in the raw data. Then, each subsequent layer builds on what the previous one has uncovered. Each layer stores what the network has learned up to that point, along with some rules it's picked up. This data processing continues through each layer until it reaches the output layer, which delivers the final result.

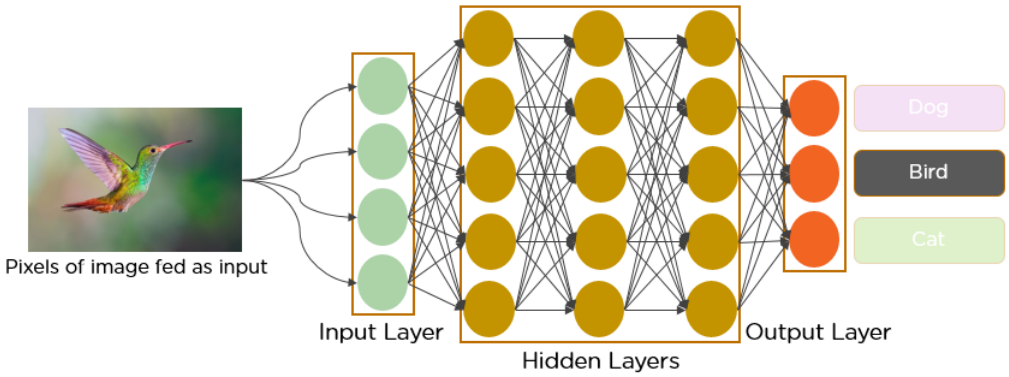

Figure 3: Deep Neural Network Structure

Training a neural network can be done in two primary ways. In supervised learning, the network is provided with clear examples of what the input should yield as output. In unsupervised learning, the network autonomously identifies patterns in the input data without explicit guidance, effectively learning on its own. For classification problems (such as classify the netowrk traffic to Benign or Abnormal) the supervised learning approach is used.

2 Experience Canvas

After a short introduction to IoT cyber security and deep learning methods, we can try to build a project case canvas follwing the template of Atlassian's Team Playbook Experience Canvas. Inspired by the classic lean canvas, an experience canvas helps clarify what problem your project is trying to solve, the customer(s) you're solving it for, and what success looks like.

2.1 Hypothesis

"The difference the team thinks the project will make for it's customers. We think that ... will have the following effect .."

We think that as more devices connect through IoT local networks, like WiFi, the need for better security will increase. This is because cyber threats, like hacking and attacks, are getting smarter. Our project's goal is to create stronger security measures specifically designed for these IoT networks. By doing this, we aim to protect the networks and the devices connected to them. This will help network providers, IoT solution makers, and regular users to keep their systems safe from cyber threats and improve the overall security of IoT networks.

2.2 Problem

"What triggered the hypothesis? Clearly list challenges, issues and root causes."

End users of IoT solution seek instant notifications to stay informed about any potential threats or anomalies on their network. This need arises because IoT networks are becoming more complex, and users want to ensure the safety and reliability of their connected devices and data. They require solutions that provide real-time visibility and proactive alerts to address emerging cybersecurity challenges in the rapidly evolving IoT landscape.

2.3 Customer personas

"Who has this problem? What motivates them? What are they trying to accomplish?"

1. **Internet Providers**: (e.g. Vodafone): They are a company that provides internet services. They want to make sure that the networks of their customers who use IoT (Internet of Things) devices, like Smart Grids, are safe from cyber threats.

2. **IoT Solution Provider**: (e.g. uni.systems): They create IoT solutions, such as Smart Grids. They want to enhance the security of their solutions to ensure they work reliably and securely for their customers.

3. **End-users**: (e.g. Quest Energy): They rely on these IoT solutions for various purposes. Their motivation is to have trust in the security of these systems, reduce worries about cyber threats, and ensure the uninterrupted operation of critical services.

2.4 Stakeholders

"Who supports this effort? Who could potentially block this project? Who also has requirements?"

1. **Network Service Providers**: (e.g. Uni4Network, Vodafone)

2. **IoT Solution Providers**: (e.g. ACS - Smart Grid, uni.systems - Software

3. **IoT Device Manufacturers**: (e.g. Cisco)

4. **End-Users**: (e.g. ACS, Public Hospitals, Government)

2.5 Team

"What experience and skills are required to set up this experience for success?"

1. Data Analyst

2. Data Scientist

3. IT developers

4. Exploitation & Marketing Manager
5. Operation Support

2.6 Value

"What is the likely user benefit and business benefit?"

1. **Network Service Providers**: These providers aim to enhance the security of their IoT networks to maintain customer trust, expand their customer base, and ensure the reliability of their services.
2. **IoT Solution Providers**: IoT solution providers are interested in improving the security and robustness of their offerings, such as Smart Grids, to meet evolving customer demands.
3. **IoT Device Manufacturers**: Manufacturers seek to enhance the security protocols of their IoT devices.
4. **End-Users**: End-users expect increased security and confidence in their IoT solutions to protect critical operations and sensitive data. They rely on robust security measures to mitigate risks.

2.7 Ideas

"What could solve the customer personas' problems and meet stakeholders' requirements?"

1. **Advanced Intrusion Detection System**: Develop an advanced intrusion detection system tailored for IoT networks. This system should employ machine learning algorithms to proactively identify and respond to emerging threats in real-time.
2. **Security Analytics Dashboard**: Create a comprehensive security analytics dashboard that provides network administrators with a centralized view of IoT device activity, anomalies, and potential security breaches. The dashboard should offer real-time monitoring and reporting capabilities.
3. **Threat Intelligence Integration**: Monitor the effectiveness of threat intelligence feeds in proactively identifying and mitigating threats. Success is the timely identification of emerging threats.

2.8 Minumum viable experience

"The smallest, easiest, fastest-to-make version of your idea that can reliably prove the hypothesis."

Advanced Intrusion Detection Sstem (IDS) will detect unusual traffic and generate reports of possible intrusions in the IoT LAN. Also, this IDS has the ability to self-train in order to be able to detect with good possibility day-0 attacks.

2.9 Metrics

"Define success metrics directly related to the desired values for this experience that will be used to prove or disprove the hypothesis."

1. **Detection Accuracy**: Measure the system's accuracy in identifying security threats. Success is defined by a high true positive rate and a low false positive rate.

2. **Response Time**: Evaluate the system's response time in reacting to security incidents. Success involves rapid detection and mitigation of threats.

3. **Reduction in Incidents**: Track the number of security incidents and breaches before and after the solution's implementation. Success is a noticeable reduction in incidents.

4. **User Satisfaction**: Gather feedback from network administrators and end-users to assess their satisfaction with the enhanced security measures. Success involves positive feedback and increased security confidence.

5. **Cost Savings**: Calculate the cost savings achieved through automated threat detection and response compared to manual security practices. Success involves significant cost reductions.

6. **Threat Intelligence Integration**: Monitor the effectiveness of threat intelligence feeds in proactively identifying and mitigating threats. Success is the timely identification of emerging threats.

2.10 End of end demo

"Tell an end to end story from the point of view of the customer that focuses on the problems solved, the solution applied and value achieved."

Imagine a scenario where Uni4Network, the network service provider, deploys the enhanced security solution in a Smart Grid environment for a customer, ACS. The demo showcases the system's capabilities:

1. **Phase 1: Deployment**: Uni4Network deploys the advanced intrusion detection system and security analytics dashboard within ACS's Smart Grid infrastructure.

2. **Phase 2: Real-Time Monitoring**: The security dashboard provides real-time monitoring of all IoT devices within the Smart Grid, including sensors, controllers, and actuators.

3. **Phase 3: Threat Detection**: The system identifies a suspicious surge in traffic on a particular segment of the network, (e.g. signaling a potential DDoS attack).

4. **Phase 4: Reporting and Analysis**: Post-incident, the system generates detailed reports on the attack, including its origin, type, and impact. This information is crucial for forensic analysis.

5. **Phase 5: Detection Mechanism Retrain**:After an incident the detection mechanism can get retrained based on the results of the Reposting and Analysis phase. This will ensure that the detection mechanism stays up to date and gain more intelligence.

3 Solution

As described in the section 2, the solution consists of an Intrusion Detection System (IDS) that uses a deep neural network to recognise patterns in the network traffic and decide if the traffic is Benign or Abnormal and also be trained enough to recognise the type of Abnormal Traffic (DoS, DDoS, Web-based etc). This solution is designed for Local Area Networks (LANs) that consist of numerous IoT devices, mainly those that can be found in a smart home installation. However this solution can also be extended to industrial IoT networks with the only condition the possibility to generate enough network traffic data in order to train the model sufficiently. This means that there is a requirement, after the installation of the network and before its production release, to perform a simulation of several known attacks to produce the required data for the training. In this case scenario, we use a dataset that we are going to describe in 3.1.2 that contains labeled network traffic of a smart home IoT network installation.

3.1 Architecture flow

To do

3.2 Development of Deep Learning Models

In this subsection we will be focusing in the Anomaly and Signature Detection parts of the solution as can be seen in the flow diagram of Figure 4. We will go thorugh the process from getting the raw data to building the final and optimal machine learning model that can be deployed in the next stage. Therefore, we will go through the standard but also experimental steps taken in the development environment of Jupyter Notebook. This aspect of development falls under the purview of a Data Scientist.

3.2.1 Deep Learning Framework

In the realm of deep learning, there are numerous frameworks available for use. The choice of a particular framework depends on factors such as:

- Project Requirements: The type of problem at hand. For instance, if you're working on natural language processing, you might prefer a framework with robust support for this type of task.

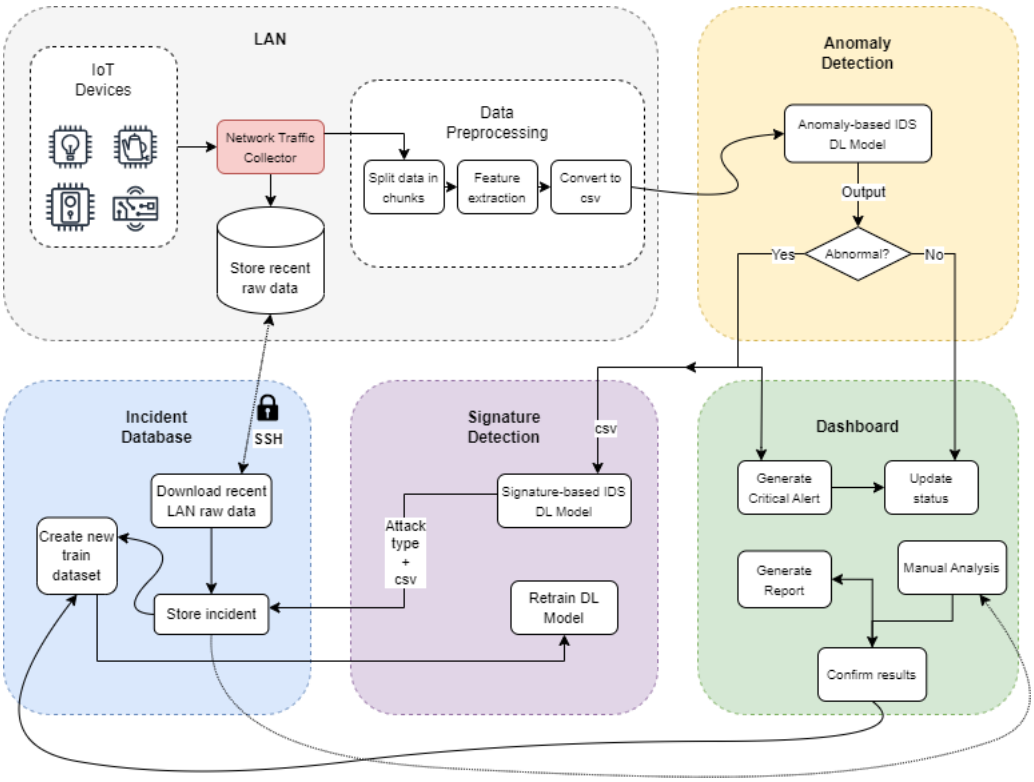

Figure 4: Architecture flow diagram

- Experience and Skills: The knowledge and experience of your team. If your team is well-versed in using Python, a framework that supports this language might be preferable.
- Computational Power and Resources: The capabilities of the available computing infrastructure can influence your choice. If your system has NVIDIA GPUs, using a framework that supports CUDA for faster model training might be a wise decision.
- Data Scale: Dealing with large datasets may lead you to choose a framework capable of distributed processing or one that is cloud-friendly.
- Community and Support: The size and activity of the framework's community can impact your decision.

Here are some of the most well-known deep learning frameworks today:

1. TensorFlow: TensorFlow is a versatile and popular deep learning framework developed by Google. It supports the development of a wide range of machine learning and deep learning applications. TensorFlow offers flexibility, scalability, and extensive support for various hardware platforms, making it a top choice for both research and production projects.

2. PyTorch: PyTorch is a Python library that supports the development of deep learning projects, including tasks in computer vision and natural language processing. It features powerful tensor computations (similar to NumPy) with GPU acceleration.

3. MxNet: MxNet is a deep learning framework that allows you to define, train, and deploy deep neural networks in a flexible range of device types, from cloud to mobile devices. It offers a user-friendly model-building interface and supports multiple languages.

4. Caffe: Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that enables users to create artificial neural networks (ANNs) with a focus on image classification. It is known for its speed and portability and is commonly used for convolutional neural network (CNN) modeling.

5. Theano: Theano is an open-source framework that allows the development, optimization, and evaluation of mathematical expressions, including efficient handling of multi-dimensional arrays. It is well-suited for large-scale computational scientific research.

Among the prominent deep learning frameworks available today, TensorFlow stands out as a versatile and popular choice. Developed by Google, TensorFlow offers a comprehensive set of tools for machine learning and deep learning applications. It excels in both research and production environments due to its flexibility, scalability, and extensive hardware support. TensorFlow's ability to seamlessly integrate with a wide range of hardware platforms makes it a top choice for developing and deploying deep learning models. Additionally, its active maintenance and updates by Google ensure its longevity and relevance in the rapidly evolving field of machine learning. These qualities, coupled with its powerful experimentation capabilities and ease of use, make TensorFlow with Keras a compelling option for researchers and practitioners alike when considering deep learning frameworks for their projects.

3.2.2 Dataset Overview

This section introduces the CICIoT2023 dataset which is used in this solution for the ML models development. It was published in 2023 in Sensors Journal from the Faculty of Computer Science, University of New Brunswick (UnB) in Canada.

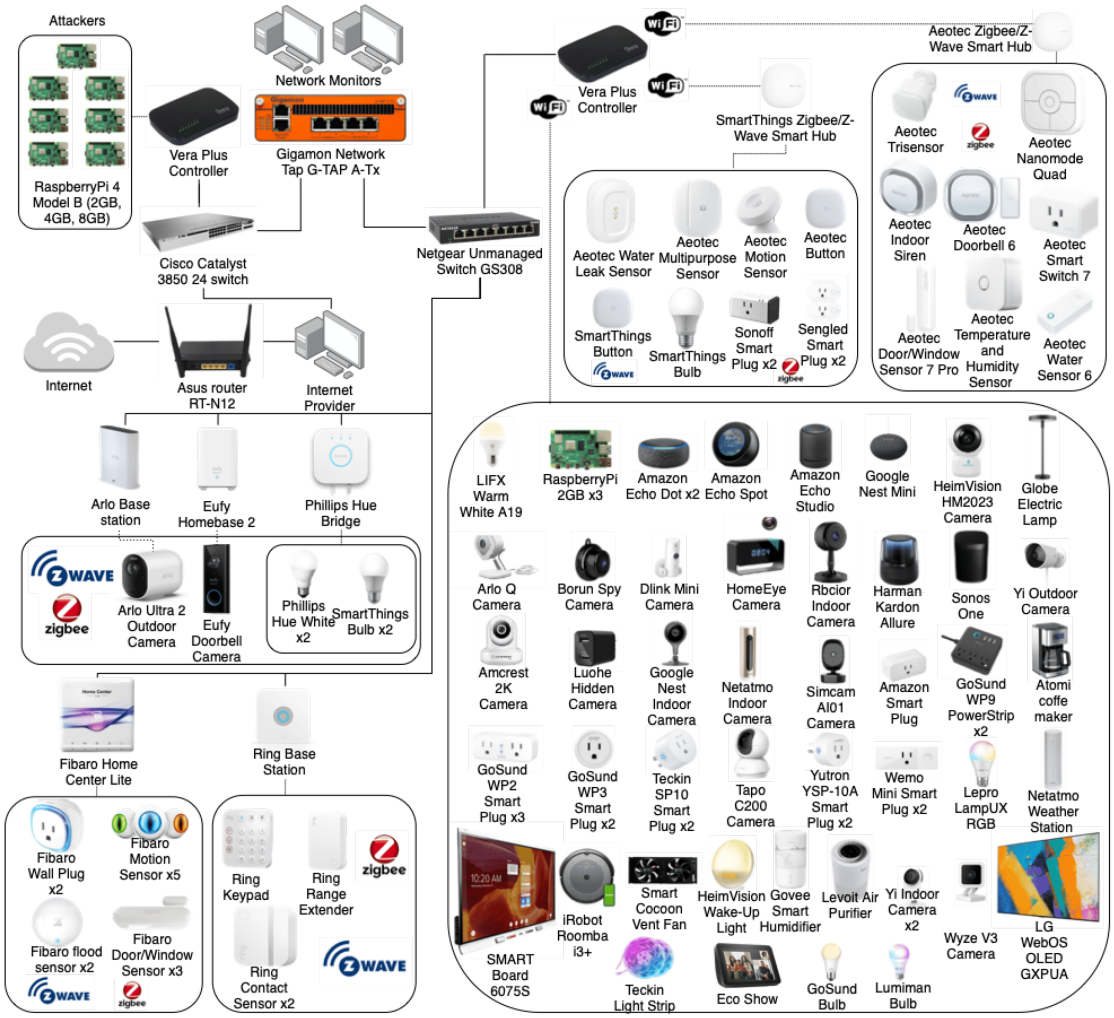

Figure 5: Network Topology

A total of 67 IoT devices were directly involved in the attacks and other 83

Zigbee and Z-wave devices were connected to five hubs. This topology mimics a real-world deployment of IoT products and services in a smart home environment. Seven Raspberry Pi devices were connected to the network via a Cisco switch and they enabled the researchers to perform several attacks to capture both benign and malicious attack traffic. Then, the Cisco switch is connected to the second part through a Gigamon Network Tap. This network device collects all the IoT traffic and sends it to two network monitors, which are responsible for storing the traffic using wireshark. In fact, a network tap is a hardware device that allow for monitoring and analyzing network traffic by connecting to a network cable and providing a copy of the traffic to other monitoring and security tools. Network taps are connected in a way so as not to affect the normal operation and provide a full-duplex, non-intrusive, and passive way of accessing network traffic, without introducing any latency or affecting the performance of the network.

The types of attacks and their data distribution is shown in figure 6. Note

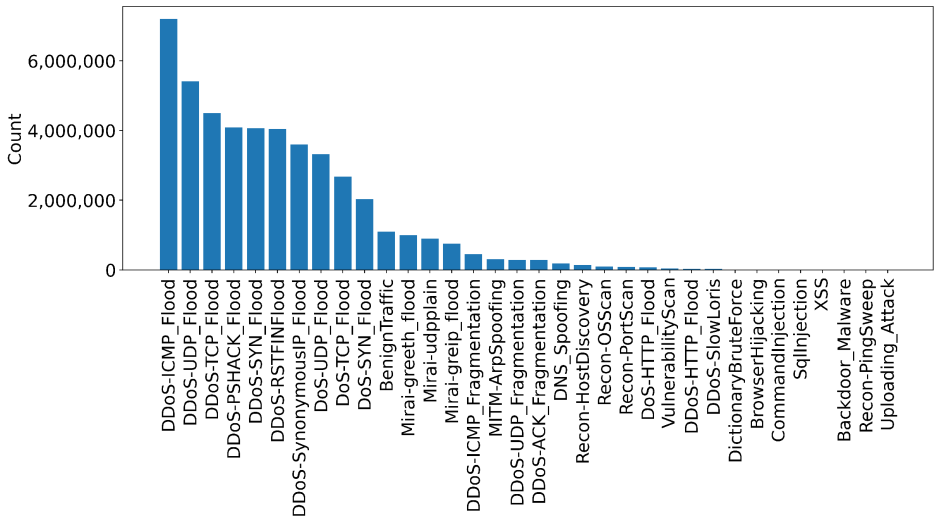

Figure 6: Dataset attack data distribution

that 3 steps are taken before the data rows are extracted; data generation, extraction, and labeling. The first phase relies on the use of different tools to execute attacks against IoT devices in the network. After that, the network traffic is captured in pcap format using Wireshark. Finally, for each attack executed, the entire traffic captured is labeled as belonging to that particular

attack. Each row in the dataset is a result of feauture extraction, using the DPKT package, of a fixed-size packet window. The features extracted for each network data flow is presented in the following table.

Features extracted from network traffic		
#	**Feature**	**Description**
1	ts	Timestamp
2	flow duration	Duration of the packet's flow
3	Header Length	Header Length
4	Protocol Type	IP, UDP, TCP, IGMP, ICMP, Unknown (Integers)
5	Duration	Time-to-Live (ttl)
6	Rate	Rate of packet transmission in a flow
7	Srate	Rate of outbound packets transmission in a flow
8	Drate	Rate of inbound packets transmission in a flow
9	fin flag number	Fin flag value
10	syn flag number	Syn flag value
11	rst flag number	Rst flag value
12	psh flag numbe	Psh flag value
13	ack flag number	Ack flag value
14	ece flag numbe	Ece flag value
15	cwr flag number	Cwr flag value
16	ack count	Number of packets with ack flag set in the same flow
17	syn count	Number of packets with syn flag set in the same flow
18	fin count	Number of packets with fin flag set in the same flow
19	urg coun	Number of packets with urg flag set in the same flow
20	rst count	Number of packets with rst flag set in the same flow
21	HTTP	Indicates if the application layer protocol is HTTP

Features extracted from network traffic		
#	**Feature**	**Description**
22	HTTPS	Indicates if the application layer protocol is HTTPS
23	DNS	Indicates if the application layer protocol is DNS
24	Telnet	Indicates if the application layer protocol is Telnet
25	SMTP	Indicates if the application layer protocol is SMTP
26	SSH	Indicates if the application layer protocol is SSH
27	IRC	Indicates if the application layer protocol is IRC
28	TCP	Indicates if the transport layer protocol is TCP
29	UDP	Indicates if the transport layer protocol is UDP
30	DHCP	Indicates if the application layer protocol is DHCP
31	ARP	Indicates if the link layer protocol is ARP
32	ICMP	Indicates if the network layer protocol is ICMP
33	IPv	Indicates if the network layer protocol is IP
34	LLC	Indicates if the link layer protocol is LLC
35	Tot sum	Summation of packets lengths in flow
36	Min	Minimum packet length in the flow
37	Max	Maximum packet length in the flow
38	AVG	Average packet length in the flow
39	Std	Standard deviation of packet length in the flow
40	Tot size	Packet's length
41	IAT	The time difference with the previous packet
42	Number	The number of packets in the flow
43	Magnitude	(Average of the lengths of incoming packets in the flow + average of the lengths of outgoing packets in the flow) 0.5

Features extracted from network traffic		
#	**Feature**	**Description**
44	Radius	(Variance of the lengths of incoming packets in the flow + variance of the lengths of outgoing packets in the flow) 0.5
45	Covariance	Covariance of the lengths of incoming and outgoing packets
46	Variance	Variance of the lengths of incoming packets in the flow / variance of the lengths of outgoing packets in the flow
47	Weight	Number of incoming packets × Number of outgoing packets

3.2.3 Data Prepossessing

Data preprocessing is a crucial step in the data preparation pipeline for machine learning and data analysis. It involves a series of operations aimed at cleaning, transforming, and organizing raw data to make it suitable for model training and analysis. This process typically includes handling missing values, dealing with outliers, scaling or normalizing features, encoding categorical variables, and splitting the data into training, validation, and test sets.

Data preprocessing ensures that the data is in a consistent and structured format, which is essential for achieving accurate and reliable results from machine learning models. Properly preprocessed data can improve model performance, reduce the risk of overfitting, and enhance the interpretability of results, ultimately leading to more informed decision-making in various applications across different domains.

1. **Handling missing values**
   In CICIoT2023 dataset no missing values (Null values) are present in any row. So we do not have to deal with this.

2. **Dealing with outliers**

3. **Scaling or normalizing features**
   Scaling typically involves transforming features to have a similar range, preventing certain variables from dominating others during model training. This is vital for the proper model's performance.

4. **Encoding categorical variables**

5. **Splitting the data**

Dealing with imbalanced data

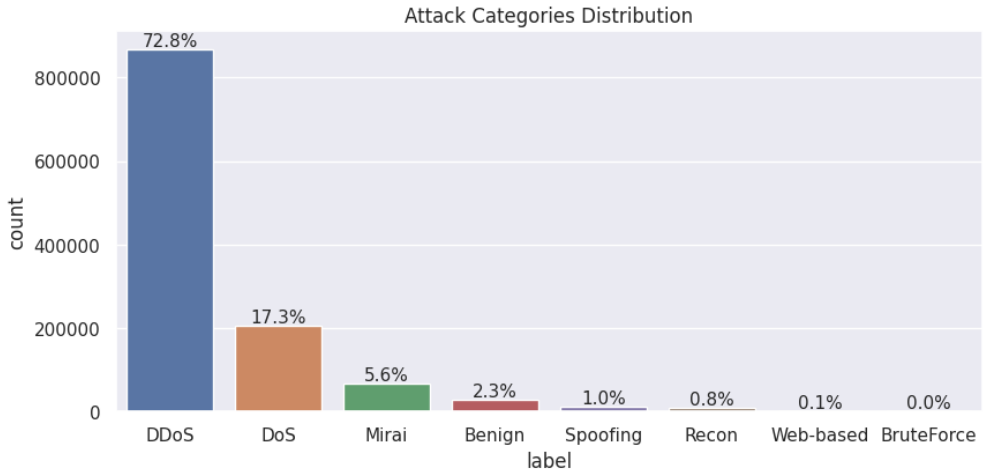

Figure 7: Attack categories distribution

3.2.4 Feature Engineering

Feature engineering is a vital aspect of data preparation in machine learning, where raw data is transformed into a more informative and relevant representation. This process involves creating new features, selecting the most relevant ones, and transforming existing ones to improve the predictive power of machine learning models.

Feature engineering often requires domain expertise to identify meaningful patterns or relationships within the data. By crafting well-designed features,

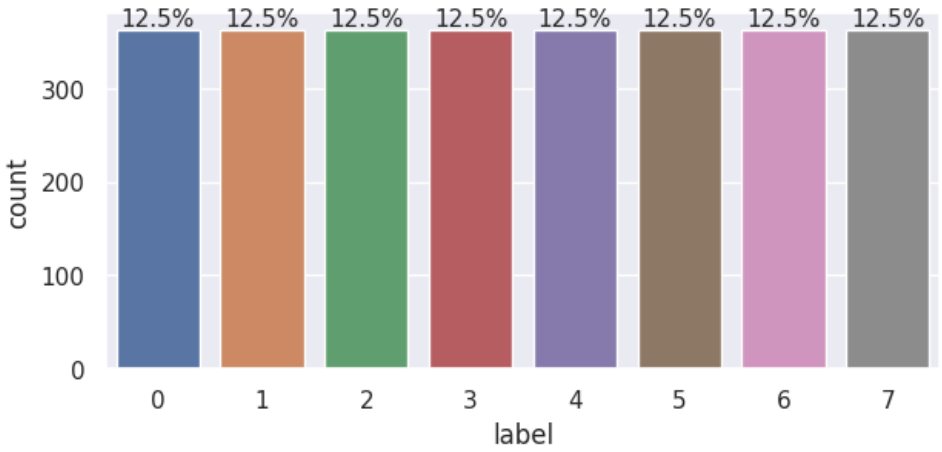

Figure 8: Attack categories distribution after undersampling

practitioners can enhance a model's ability to capture underlying patterns, increase its robustness to noisy data, and accelerate convergence during training. Feature engineering is a creative and iterative process that plays a pivotal role in the success of machine learning projects, as the quality and relevance of features can significantly influence model performance and the ability to extract valuable insights from data.

3.2.5 Model Design

Model design is a pivotal phase in machine learning where the architecture and structure of the model are crafted to address a specific problem. It involves selecting the appropriate type of model, like convolutional neural networks (CNNs) for images or recurrent neural networks (RNNs) for sequences, and making decisions about the number of layers, types of layers, activation functions, and connectivity between neurons. The objective is to create a model that effectively captures the underlying patterns in the data while balancing complexity with interpretability. Model design requires domain expertise, experimentation, and iterative refinement to achieve optimal model performance. It plays a crucial role in determining the model's ability to extract meaningful insights or make accurate predictions from data, making it a cornerstone of machine learning success.

ANN

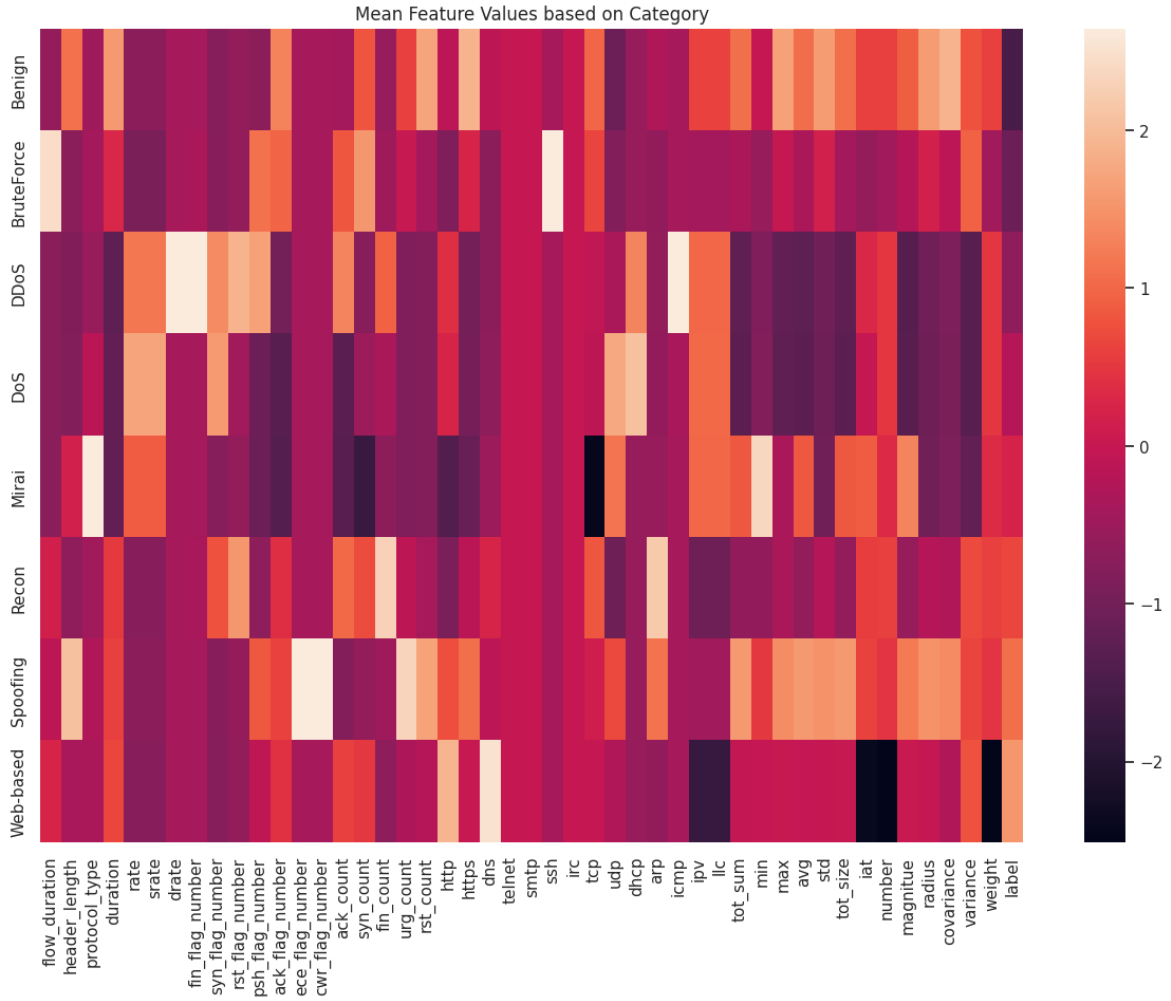

Figure 9: Mean of features values comparison

CNN

3.2.6 Model Compilation

Model compilation is a pivotal step in the machine learning workflow, where the architecture of the model, its training process, and its evaluation metrics are defined. During this phase, practitioners specify crucial components such as the loss function, optimization algorithm, and evaluation metrics to tailor

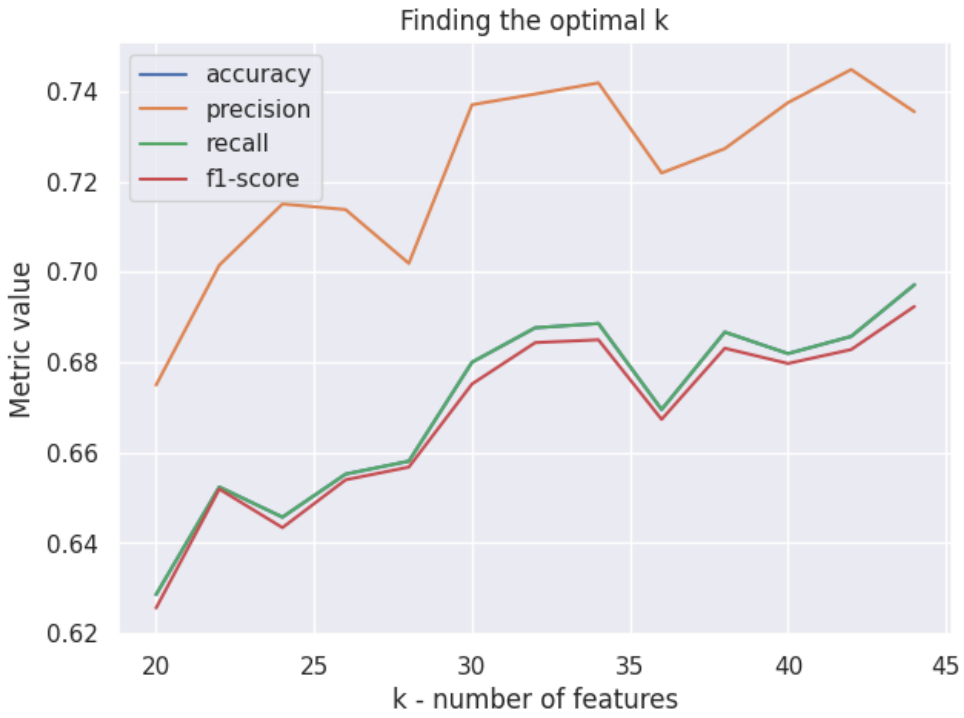

Figure 10: Selecting optimal k features

the model to a specific task.

The choice of loss function is central, as it quantifies the model's performance and guides its learning process. Optimization algorithms like stochastic gradient descent (SGD), Adam, or RMSprop dictate how the model's parameters are updated during training to minimize the chosen loss. Additionally, specifying evaluation metrics ensures that the model's performance can be effectively assessed during and after training. Model compilation requires a thoughtful understanding of the problem and often involves hyperparameter tuning to optimize model performance. Ultimately, the compiled model is a crucial building block in the development of machine learning solutions, shaping its learning process and guiding its ability to make predictions or classifications.

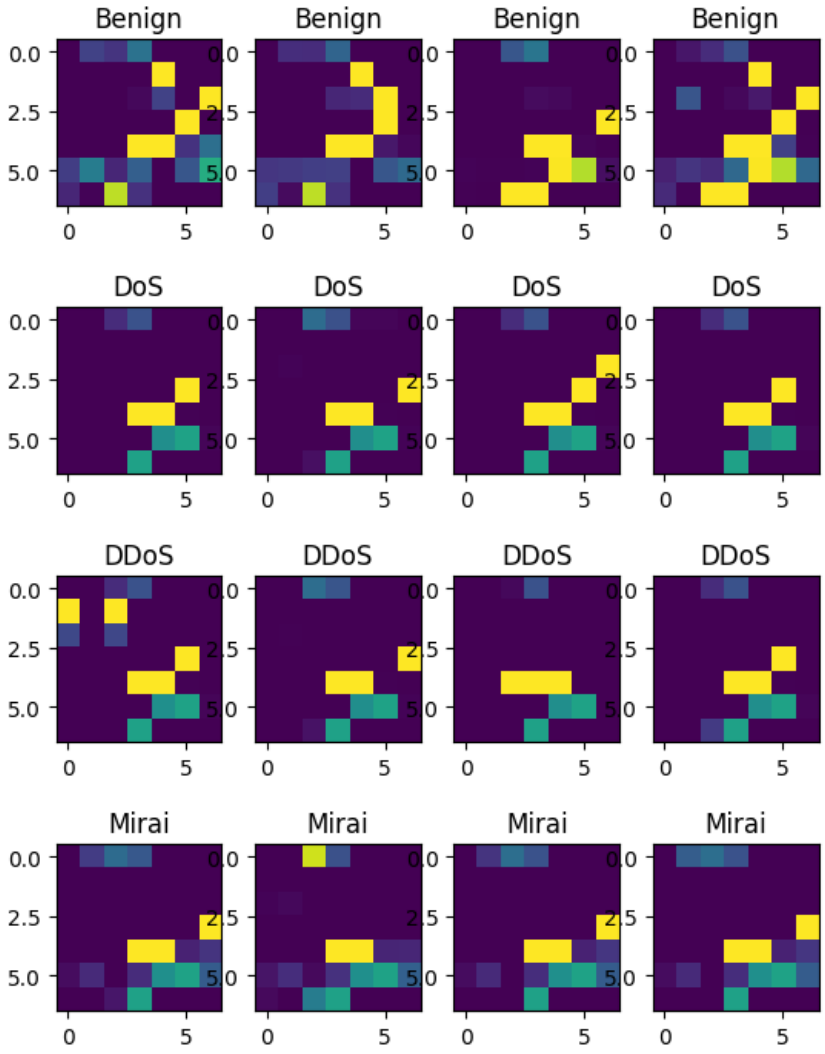

Figure 11: Visualizing Traffic

3.2.7 Model Training

Model training is a fundamental stage in the development of machine learning models, during which the model learns patterns and relationships within the training data. This process involves iteratively adjusting the model's parameters using optimization algorithms to minimize a specified loss function. The goal is to find the optimal parameter values that enable the model

to make accurate predictions or classifications. Training data is typically divided into mini-batches, and the model's parameters are updated using gradient descent-based algorithms like stochastic gradient descent (SGD) or Adam. The training process continues until a convergence criterion is met or for a predetermined number of iterations. Model training requires careful selection of hyperparameters, monitoring for issues like overfitting, and validation to ensure that the model generalizes well to unseen data. The trained model's performance is evaluated using test data, and it can then be used for making predictions or solving real-world problems.

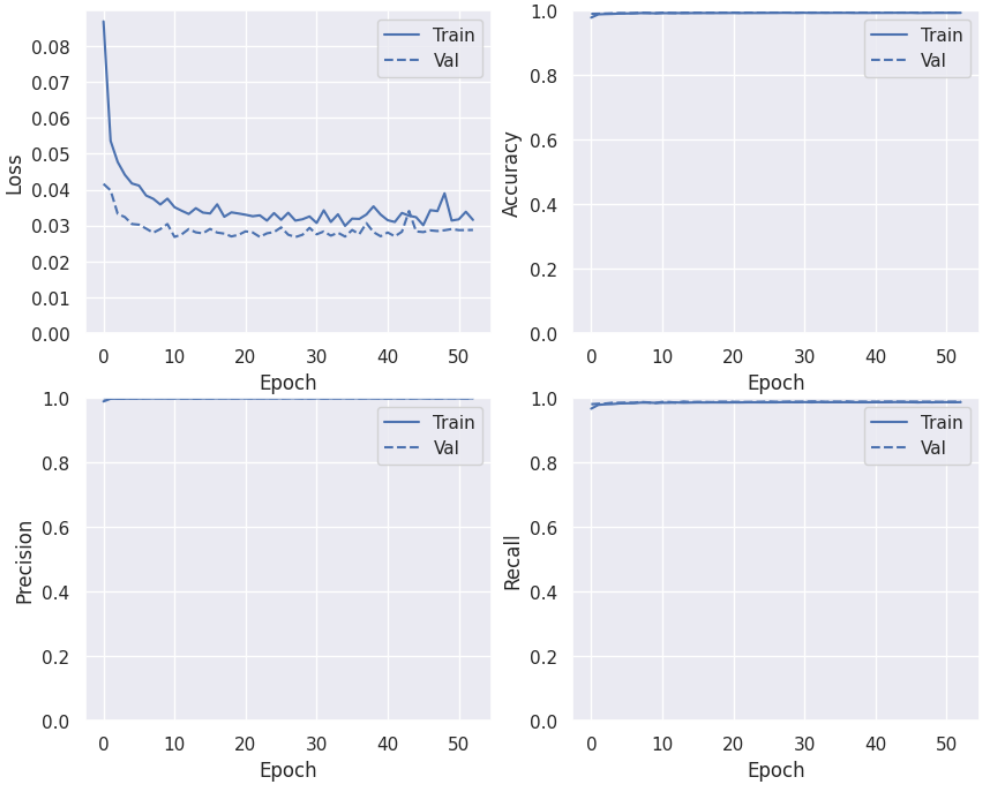

Figure 12: Training of Binary Classification Model

3.2.8 Model Hypertuning

Model hyperparameter tuning, often referred to as hyperparameter optimization, is the process of systematically searching for the best combination of

hyperparameters to optimize a machine learning model's performance. These hyperparameters, such as learning rate, batch size, and the number of layers or units in a neural network, are settings that are not learned from the data but need to be set before training. Hyperparameter tuning involves techniques like grid search, random search, or Bayesian optimization to explore the hyperparameter space efficiently.

It plays a crucial role in fine-tuning a model, improving its generalization, and achieving the best possible results on a given task. Effective hyperparameter tuning can make the difference between a model that underperforms and one that excels in solving complex problems.

3.2.9 Regularization

Model regularization is a set of techniques used in machine learning and deep learning to prevent overfitting and improve the generalization performance of models. Overfitting occurs when a model learns to fit the training data too closely, capturing noise and irrelevant patterns rather than the underlying structure. Regularization methods introduce constraints or penalties on the model's parameters during training, discouraging it from becoming overly complex. Common regularization techniques include L1 regularization (Lasso), L2 regularization (Ridge), and dropout for neural networks.

These methods encourage the model to be simpler and help it generalize better to unseen data, ultimately improving its predictive accuracy and robustness. Regularization is a critical component of model training and plays a crucial role in achieving models that perform well on diverse datasets.

3.2.10 Model evaluation

Model evaluation is the process of assessing the performance of a machine learning or statistical model on a dataset. It involves comparing the model's predictions or classifications to the actual ground truth to measure how well the model generalizes to unseen data. Evaluation metrics, such as accuracy, precision, recall, F1-score, mean squared error, or others depending on the problem type, provide quantitative measures of a model's performance. The choice of the appropriate evaluation metric depends on the specific problem and the desired model outcomes.

Model evaluation helps practitioners determine the model's effectiveness, identify areas for improvement, and make informed decisions about deploying the model in real-world applications. It is a crucial step in the machine learning workflow, ensuring that the model meets the desired criteria for reliability and accuracy.

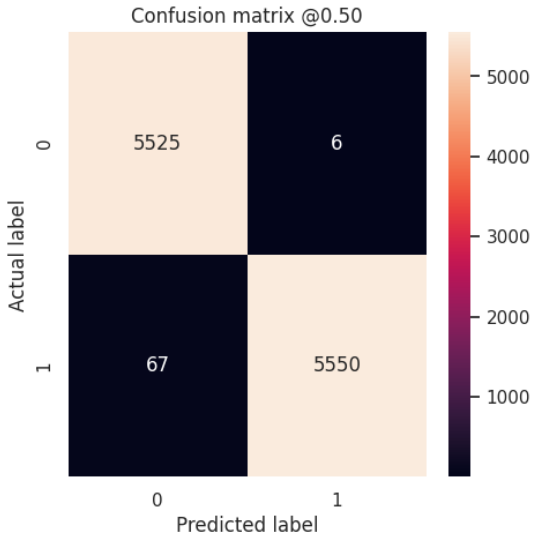

Figure 13: Confusion matrix for binary classification

3.3 Deployment of Models

To do

3.3.1 ML Flow

To do

	precision	recall	f1-score	support
Benign	0.64	0.75	0.69	148
BruteForce	0.55	0.52	0.54	130
DDoS	1.00	0.45	0.62	139
DoS	0.62	0.99	0.76	122
Mirai	1.00	0.98	0.99	121
Recon	0.78	0.48	0.59	108
Spoofing	0.86	0.56	0.68	144
Web-based	0.47	0.74	0.58	138
accuracy			0.68	1050
macro avg	0.74	0.69	0.68	1050
weighted avg	0.74	0.68	0.68	1050

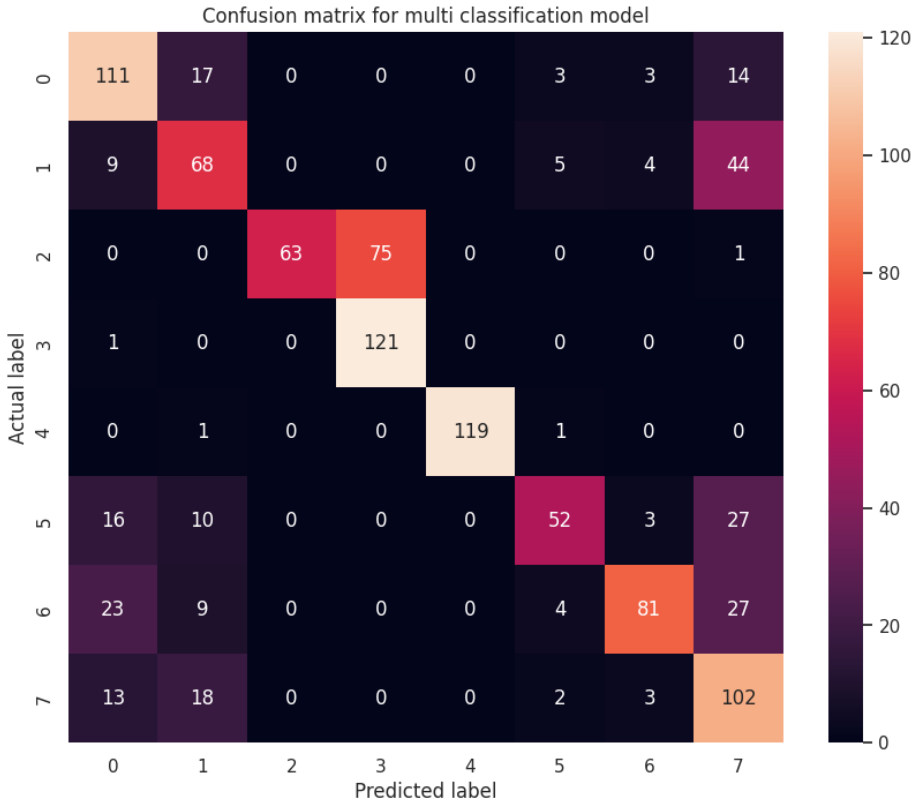

Figure 14: Confusion matrix for multi classification

4 Message Queuing Telemetry Transport

4.1 The protocol

The MQTT stands for Message Queuing Telemetry Transport and it is a messaging protocol for restricted low-bandwidth networks and extremely high-latency IoT devices. This makes it an ideal protocol for machine-to-machine (M2M) communication.

MQTT works on the publisher / subscriber principle and is operated via a central entity called "broker". This means that the sender (publisher) and the receiver (subscriber) have no direct connection. The data sources report their data with a publish action an all recipients with interest in certain messages (called "topics") get the data delivered because they have registered as subscribers.

This protcol is very often used in IoT and IIoT (Industrial IoT) all the way to connecting cloud environments. For example, in a smart grid, sensors can be considered as publishers in the MQTT protocol because they report their status by publishing messages to a certain topic. A web-client, can monitor the sensors' values from a web-interface, are considered as clients who are subscribed to the topics of sensor devices.

4.2 MQTT Security

Security in MQTT is structured into multiple layers, each designed to mitigate specific types of threats. MQTT aims to offer a lightweight and user-friendly communication protocol tailored for IoT (Internet of Things) applications. While the core MQTT protocol specifies a limited set of security mechanisms, MQTT implementations often rely on established security standards, such as SSL/TLS, to enhance security. Leveraging widely accepted security standards is a pragmatic approach, given the complexities associated with ensuring robust security. Here is a concise overview of the security layers within MQTT:

1. **Network Level**:
   For establishing a secure and trustworthy connection in MQTT, one

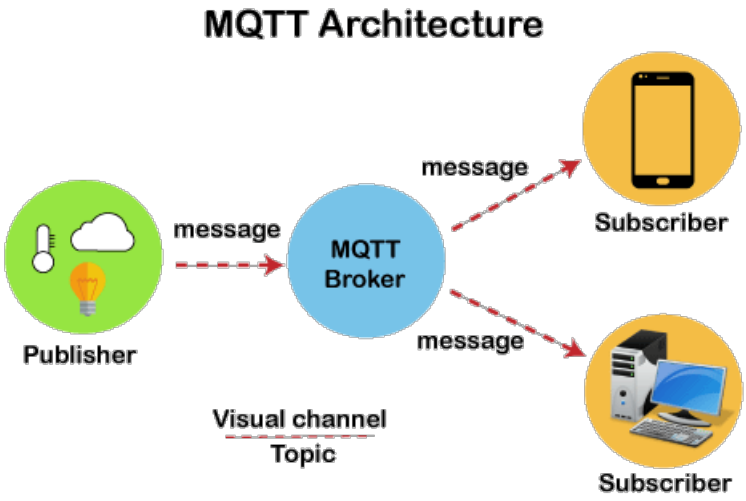

Figure 15: MQTT Protocol

approach involves utilizing a physically secure network or a Virtual Private Network (VPN) for all communication between MQTT clients and brokers. This approach is particularly well-suited for gateway applications where the gateway acts as an intermediary, connecting to devices on one side and to the MQTT broker via a VPN on the other side. This physical or virtual isolation helps protect communication channels from external threats and eavesdropping.

2. **Transport Level**:
   When the primary objective is ensuring data confidentiality during transmission, the industry-standard TLS/SSL (Transport Layer Security/Secure Sockets Layer) protocols are commonly employed for transport encryption in MQTT. TLS/SSL provides a proven and secure method to prevent unauthorized access to data in transit. It also offers a robust mechanism for client-certificate authentication, verifying the identities of both the MQTT clients and the broker. The feasibility of implementing TLS on resource-constrained devices is discussed in greater detail in a separate post.

3. **Application Level**:
   At the transport level, MQTT communication is secured through en-

cryption, and client identities are authenticated. The MQTT protocol itself offers built-in mechanisms for client identification and the use of username/password credentials to authenticate devices at the application level. The authorization and control of what each device is allowed to do are typically determined by the specific broker implementation, providing fine-grained access control. Furthermore, it's possible to implement payload encryption at the application level, ensuring the security of transmitted information without necessitating full-fledged transport-level encryption.

By leveraging these various levels of security—network, transport, and application—MQTT is equipped to address diverse security requirements, ensuring the integrity, confidentiality, and trustworthiness of IoT and M2M communication. This multi-layered security approach enables MQTT to cater to a wide range of use cases, from constrained IoT devices to more robust and demanding applications.