# Malicious Data Classification in Packet Data Network through Hybrid Meta Deep Learning

by

Sakib Uddin Tapu
18301271
Samira Afrin Alam Shopnil
18301076
Rabeya Bosri Tamanna
18301188

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2023

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

Sakib Uddin Tapu
18301271

Samira Afrin Alam Shopnil
18301076

Rabeya Bosri Tamanna
18301188

# Approval

The thesis titled "Malicious Data Classification in Packet Data Network Through Hybrid Meta Deep Learning" submitted by

1. Sakib Uddin Tapu (18301271)
2. Samira Afrin Alam Shopnil (18301076)
3. Rabeya Bosri Tamanna (18301188)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 26, 2023.

**Examining Committee:**

Supervisor:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

Advancements in wireless network technology have provided a powerful tool to boost productivity and serve a strong communication which overcomes the limitations of wired networks. However, because of using wireless networks, security is an increasing concern among the community. At the time of our study, we are in the era of 5G networks. Although we are in the 5th generation of telecommunication we are still struggling with security. The upcoming generation, 6G, aims to solve the security concerns by providing a secure and trust networking system. In our study, we aim to integrate AI and more advanced infrastructure which will provide a tremendous solution in this regard. In order to deal with this issue we primarily aim to come up with a solution that provides a reliable intrusion detection system in spite of being trained with a small amount of data. In our study, we aim to integrate AI and more advanced infrastructure which will provide a tremendous solution in this regard. Thus, we employed a trusted networking system based on AI. Here, at first we primarily focused on Reinforcement Learning (RL) to classify the network data coming from the untrusted packet data networks (PDN), whether it is malicious or not. Another existing problem is people currently rely on machine learning techniques to create a trustworthy networking system. However, it hinders the development of getting a reliable network as the number of real publicly available malicious data is not sufficient to train a model properly and in real life people are not very keen to share these data as they are sensitive. Therefore, we propose a novel idea of hybrid meta learning in the detection of malicious packet data. We use a combination of Siamese and Prototypical network where Siamese network is used for binary classification and Prototypical network is used for multi class classification. As both approaches are based on meta learning techniques, it requires a very small amount of data. By utilizing this characteristic of meta learning, we were able to train our model with just 3000 data samples and achieve more than 90% accuracy for both meta learning tactics. Lastly we provide a comprehensive study on the given RL methods and hybrid meta learning and share our future thoughts. The purpose of our study is to provide a secure and trustworthy network domain which enhances the communication between end users.

**Keywords:** Reinforcement learning, A2C, PPO, meta-learning, few-shot-learning, siamese-network, prototypical-network,intrusion-detection, malicious-data-classification, CSE-CIC-IDS2017, CSE-CIC-IDS2018

# Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Md. Golam Robiul Alam sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

# Table of Contents

# List of Figures

# List of Tables

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

| | |
|---|---|
| A2C | Advantage Actor Critic Algorithm |
| CNN | Convolutional Neural Network |
| FL | Federated Learning |
| ML | Machine Learning |
| PDN | Packet Data Network |
| PPO | Proximal Policy Optimization Algorithm |
| RL | Reinforcement Learning |

# Chapter 1

# Introduction

## 1.1 Research Motivation

Communication has always been a fundamental part of life for human interaction. People were always looking for an efficient and effective communication medium. Following that legacy, we entered into wireless communication from wired. Now-a-days the emphasis on wireless technology is progressing at a rapid pace. While enjoying the benefits of wireless communication, we also face malicious attacks which are really alarming. As the cyber attacks are evolving, attackers are exploiting our system with a lot of vulnerabilities which may hamper our system on a large scale. Here our motivation for this work is to contribute to the development of the up-coming network infrastructure. Combination of meta learning approaches could be a tremendous solution in this regard. Moreover, at this moment, the majority of conventional machine learning solutions for intrusion detection are heavily reliant on a larger number of samples and most of them are unable to identify any unknown threat in the new challenging network environment. Thus the attackers can easily bypass the security of the system with an advanced behavior to accomplish their objective. Therefore, researchers are currently working to develop a method that can recognize existing assaults from their patterns and foretell other attacks that they have never encountered. Additionally, because network data is extremely confidential and not accessible to the general public, there is currently a requirement for a methodical strategy for anomaly detection with a limited amount of attack samples [16]. Furthermore, with the substantial development of the network, the number of attacks are increasing and their patterns are also changing. In this case, anomaly detection becomes harder when a model faces any abnormal or undefined behavior for the very first time. Developments to date have highlighted two dominant challenges. The first could be a robust model that can be trained with a minimum number of instances and still can predict any threat and the other one could be, the model should be able to protect the system from the emergence of any abnormal unknown behavior. We believe that our study will give a new edge in terms of establishing a communication system using Hybrid Meta Deep Learning technology that will substantiate a trusted network where users will connect in a more secure and convenient way.

## 1.2 Problem Statement

At the time of our research study, we are still struggling with network security. Therefore, scientists are working to develop a remedy that can somewhat mitigate this issue. But the procedure was everything but simple. Malicious network data is scarce. As a result, datasets that are made publicly available are highly unbalanced.Moreover, most of the existing research is based on machine learning techniques which require a lot of data. Thus it is a very difficult task to build an effective model with the publicly available unbalanced dataset. Furthermore, having to train on large amounts of data also requires a lot more data processing which in turn increases the time to train the model. Moreover, all the existing works are only able to identify 8 distinct classes at most. People also show unwillingness in sharing their data because it is very sensitive and confidential. Therefore, it is difficult for researchers to obtain new real traffic data and develop better models. Number of malicious data in the publicly available dataset is insufficient and the majority of machine learning models need to be trained with a lot of data. Thus, people are trying to build models that can be trained with very small amounts of data and yet work efficiently. There is also a lack of meta learning approaches being used for network security, specially hybrid models that provide advantages of different techniques.

## 1.3 Contributions

The focus of our study is to propose a trusted platform between users that allows them to communicate with the highest level of information security in a more appropriate fashion than current state-of-the-art networks. In this work, we aim to address an existing problem which is introducing models that not only identifies the malicious attack in the packet data flow but also classifies them into multiple classes with minimal amount of data as the number of publicly available dataset for malicious data is very limited. Moreover, the state-of-the-art methods can distinguish the highest 8 different classes with a large number of samples. Thus we took the initiative to solve the challenges to detect more malicious attacks with smaller datasets.
The contributions of the paper are summarized as follows:

- We proposed a Hybrid Meta Deep Learning based approach for malicious data classification which is able to classify 15 distinct labels with greater than 90% accuracy and F-1 score while being trained with only 3000 data samples
- We built a reinforcement learning based approach with A2C and PPO. These are two different frameworks with the same ideology. With these two approaches, we could achieve approximately 70% accuracy
- In the hybrid meta deep learning technique, the Prototypical network and the Siamese network are combined in order to construct a hybrid meta learning-based model that will ensure secure user communication. Meta learning approaches' characteristics enable improved accuracy with smaller data samples
- At the time of our research, we were unable to locate any papers using hybrid meta learning in the field of network security. In order to address the scarcity

of malicious network traffic in a network environment, we took the initiative to introduce a hybrid meta learning method for intrusion detection

- We present several analyses to demonstrate the efficacy of our method for identifying 15 different classes using CSE-CIC-IDS17 and CSE-CIC-IDS18 datasets
- We provide a comparative analysis between our previous RL approach and hybrid meta learning approach and present a convenient interpretation of our research with some of the current state-of-the-art intrusion detection models to show the effectiveness of Hybrid Meta Deep Learning

# Chapter 2

# Literature Review

## 2.1 Related Work

Due to the trivial amount of literature related to reinforcement learning and meta learning in the cyber security domain, it is beyond the scope of this paper to provide an extensive background of affiliated work. However, there will be overwhelming evidence that a trusted platform is needed for PDN/network data. Furthermore, there will be ample proof that a trusted platform can be achieved by hybrid meta learning methods with a limited amount of samples to send packet data.The following literature review will briefly describe the fact.
In [9], Niknam et al. demonstrate the importance of FL in wireless networks. As ML are data-driven applications, all data cannot be shared due to privacy concerns. Thus due to data insufficiency, ML models cannot provide better results. In this regard, this article demonstrates an approach in wireless networks using FL, which will overcome the limitations of the ML mechanism. In FL, the models are trained in the local devices where data is generated. In this approach, the models are constantly trained by the new data, and users do not need to compromise their privacy in any way.
In this study [12], the researcher analyzed the concept of Customer Edge Switching (CES) in the context of 6G and compared it with the ITU-T framework. CES is a trusted SDN-based framework with a clear policy and effective tools for building a system effective policy, which is ambiguous in the ITU-T framework. They emphasize the importance of having a trustworthy network that can manage all types of traffic and security by shifting security-related processing to the cloud, where security will be assured by security intelligence, and security updates will be deployed quickly.
The study in [11] describes a network architecture that leverages Blockchain innovations to enhance the reliability of routing data. They use RL in the architecture mentioned above to assist the routing node in choosing the next appropriate routing node. Their trusted routing data management system is based on the PoA Blockchain, enabling collaboration on routing transactions across all routing nodes and server nodes.This design permits the learning model to select the optimal routing connection for normal nodes, which prevents single point attacks and maintains transaction traceability and routing information sources.Then, they examine the suggested trusted routing scheme's security. Their architecture provides 78% delay reduction when compared to the BP (Backpressure) algorithm, 52% when compared

to the QL-BP, and 67% when compared to the TB-BP (Trust-based backpressure) algorithm. Their RLBC algorithm works excellent when there are 25% malicious nodes in the routing environment.
In [15] Y. Liu et al. proposed an advanced system that uses a variety of emerging technologies or approaches to provide communication-efficient, safe, and privacy-enhanced FL. They discussed communication-efficient FL from the system-level and algorithm-level aspects, paving the way for wider-scale FL deployment to be used in 6G communications. Unfortunately, their FL mechanism would not do anything to resist or mitigate these malicious threats from a system standpoint. They also stated that to achieve human-centric communication services in 6G, creating an interpretable FL model is required. Furthermore, they discussed the importance of establishing a quality-based approach to incentivize more people to participate in FL and receive more significant rewards for providing high-quality data.
In [14], the author designed a unique incentive system to encourage additional data owners to participate in the FL process while maintaining privacy. The FL platform's incentive and privacy challenges are examined in this paper. To implement their approach, they used Mechanism design (MD), Differential privacy (DP), and Vickrey-Clarke-Groves (VCG) mechanisms. Compared to others, their DP-based VCG technique can ensure a higher profit. It also achieves suitable performance stability in the FL system platform, exceeding the current HIFL, SIFL, and FLPA systems. Unfortunately, their proposed technique in mobile crowdsensing does not allow for collaborative ML among various model owners.
In [18],W. Zhijun et al. primarily concentrate on low-rate DoS attacks, their mechanism, and make an effort to determine the LDoS attack generation concept. Additionally, they categorize LDoS attacks and current defense strategies based on the time and frequency bands in which detection and defense are carried out, and they emphasize the filter approach to defense against LDoS attack. By doing rigorous analysis, they figured out that feature detection approaches are more suited to identifying LDoS assaults due to the high positive rate. It is also simpler to put into practice and practical. Most importantly they make an effort to take the lead in motivating the researchers to investigate efficient techniques to identify and mitigate LDoS assaults.
In order to categorize the obstacles with the existing security models and to generate new directions for security framework developments using effective ML or DL methods, P. L. S. Jayalaxmi et al. provided useful information in [24] for industry and academia. These methods minimize the need for human intervention and quickly automate detection. Additionally, they discussed the significance of several Artificial Intelligence (AI)-based methodologies, tools, and methods utilized in Internet of Things (IoT) detection and/or prevention systems. Furthermore, they built a strong foundation for a prediction model and offered a blueprint for a mapping technique for analyzing the level of risk. Additionally, they suggested an integrated multilevel hybrid architecture that recognizes all varieties of security threats for future development by combining signature and anomaly detection with risk factor mapping.
In [16] K.Shaukat et al. discussed about a concise overview of machine learning techniques and how they have been or may be used to identify and categorize cyberattacks such intrusion detection, vulnerability scanning, and spam filtering on mobile and smartphone devices as well as computer networks. They also discussed

ML's shortcomings in the cyber security domain. Some of the difficulties include the following: ML techniques require a significant quantity of high-performance resources and data while training the models, one ML model cannot effectively identify different security assaults, and early attack prevention is another significant challenge.
In [27] this study, S. Neupane et al. examined the current state of explainable AI (XAI) for IDS, its limitations, and how these challenges relate to the creation of an X-IDS. In particular, the black box and white box approaches were thoroughly covered by them. The black box approach necessitates post-hoc explanation techniques to make the assumptions more comprehensible, whereas the white box approach makes the model in use naturally comprehensible. In terms of their effectiveness and capacity to generate explanations, they also discussed the tradeoffs between these techniques. In addition, they recommended using a three-layered generic architecture as a reference when creating an X-IDS.
In [25] this survey, S Y.Khamaiseh et al. conducted a thorough analysis of the adversarial attack techniques and how they function. They also presented a rigorous analysis of the attack technologies. They also include a detailed description of the most modern defense techniques and their reliability against hostile attacks. Last but not least, they emphasized future research directions as well as the ongoing challenges and unresolved concerns in this area. In order to determine the acceptability of the current state of the art technique as well as the incoming mechanism, multiple approaches should be implemented in the research on adversarial attacks and carry out various evaluation strategies.
In [21] M Wazid et al. concerned on the 5G network system and other allied fields to offer a thorough analysis for the domain's future developments. In the context of the 5G network environment, they addressed various system types, security requirements, potential attacks, and security protocols. Some of the difficulties they cited include the need for protocols to simultaneously defend against multiple attacks, the need for low computation power, low communication costs, and small storage sizes without compromising system security, the need for protocols to operate in complex environments while maintaining high scalability, the need for protocols to support a variety of devices and the mechanisms connected to them, and others. The primary objective of this research is to compile comparisons of the security protocols currently in practice and the ways to tackle the 5G-enabled IoT under one roof for the benefit of upcoming researchers.
By using MultiBoosting multi classifiers, in [3], R. Bie et al. emphasized the significance of a meta learning-based approach for network intrusion detection that greatly improves upon the detection performance of conventional machine learning intrusion detection methods. Additionally, this research presented a method known as Symmetrical Uncertainty (SU) that aids in reducing the features in network connections, hence enhancing detection precision.
In [13] J. Kim et al. introduced a deep learning based intrusion detection model especially for the identification of denial of service or DoS attacks. In their paper, they used CNN for binary and multiclass classification and gave a comparison between CNN and RNN and provided a comparative analysis of the hyper-parameters that can produce excellent outcomes. They evaluated their model compiling all the DoS attacks from CSE-CIC-IDS 2018 and KDD CUP 1999. Although they achieved a decent accuracy with their proposed model, it should be emphasized that the

datasets they used also contain a variety of malicious attacks and those were not taken into account. Moreover, they used 10407862 samples to reach their accuracy. In [23] H. Hindy et al. introduced a Siamese Network model that is employed as the One-Shot learning architecture to classify cyber attacks. Moreover, the network's performance in identifying a new cyber-attack class without retraining is evaluated where there are relatively few newly labeled attack classes. However, they only considered 5 types of attacks present in the dataset.
In order to mitigate imbalance for risk prediction, in [10] D. Sun et al. presented a Siamese Network Classification Framework (SNCF) that can transform the Siamese network to a classification based on resemblance. Additionally, SNCF exhibits high performance on feature dimensionality minimization. Moreover it is independent of feature engineering, and is insensitive to data distortion. They demonstrated that their proposed configuration is more scalable, more effective than others, and less linearly dispersive in an unbalanced dataset.
In [8] J. Kim et al. illustrated deep-learning approaches and constructed a convolutional neural network (CNN) model for intrusion detection. Furthermore, they showed a comparative study with RNN and the experimental result reflected that CNN outperforms RNN. However, instead of incorporating the complete dataset, they used each subset of the CSE-CIC-IDS 2018 dataset that was based on a particular day and presented results for each subset where each of them contained up to three different types of malicious attacks along with benign or non-malicious behavior.
With the use of a few-shot learning approach, in [19] D. Park et al. developed a Siamese Convolutional Neural Network (Siamese-CNN), which demonstrates great outcomes with only a minimal amount of training data. Additionally, utilizing the Leipzig Intrusion Detection Data Set (LID-DS), a host-based intrusion detection model including pre-processing, vector-to-image processing, training, and testing stages is designed to analyze and enhance the effectiveness of the system. They compare Vanilla Convolutional Neural Network (Vanilla-CNN) to Siamese-CNN as well, demonstrating the superior performance of their suggested approach.
A few-shot learning-based Siamese capsule network was created by Z. Wang et al. in [20] to address the lack of training data for anomalous network traffic and improve the detection of suspicious threats. In addition, the Siamese network is effectively incorporated with an unsupervised sub type sampling technique to enhance the detection of network intrusion attempts in the case of unbalanced training data. Their experimental findings demonstrate that they have successfully detected both known attacks and unknown attacks with a significant accurate classification rate utilizing a relatively small number of samples. They intend to add parallelization mechanisms in the future to boost the method's detection effectiveness and make it more applicable to real-world intrusion detection applications.
By incorporating a prototype module into a Siamese network, the author in [17] J. Wang et al. discussed a few shot learning architecture that uses the Euclidean distance to learn high quality prototype representations of each class. By performing image classification they claimed that the suggested architecture can assist the model in generalizing to new classes not included in the training set, despite only several samples of each class.
A Siamese-prototype network with prototype self-calibration and inter-calibration for few-shot remote sensing image classification was discussed in [22] by G. Cheng et

al. To get suitable prototypes, they first calibrate the ones produced from support features using the supervision knowledge from support labels. Then, they take into account how confidence scores interact between the support and query samples to further calibrate the prototype. However, their model struggles when dealing with enormously challenging samples.
To increase the semantic discriminability between prototypes, in [26] S. Mo et al. suggested a concise contrastive learning approach that employs metric loss in the Siamese style.They carried out comprehensive evaluations on numerous benchmarks, and the results depict the effectiveness of visual presentation for image classification.

## 2.2 Background Study

This section offers useful supporting information for comprehending how our suggested architecture operates. This will provide a preliminary idea to the readers to understand our work in a convenient way.

### 2.2.1 Reinforcement Learning

Reinforcement Learning is a machine learning approach which teaches an agent how to operate in a certain environment by observing how actions affect the environment. Good actions result in positive feedback for the agent, whereas each bad action results in negative feedback or a penalty. The objective is to learn the best behavior in an environment that maximizes the reward. Similar to how children explore their surroundings and discover the behaviors that enable them to accomplish a task, the optimal behavior is learnt via interactions with the environment and observations of how it responds. Exploration and exploitation are properly balanced in reinforcement learning algorithms. The exploration phase is one of the primary advantages of RL-based techniques. Essentially, this prevents optimizers from getting stuck in a local optimum, allowing it to find a global optimum more quickly. This process is akin to a trial-and-error search. Both immediate and future rewards are taken into account when determining the quality of actions. Because it can discover the actions that result in success in an unobserved environment without supervision, reinforcement learning is a particularly effective algorithm. Most importantly, Reinforcement learning algorithms are preferable mechanisms for finding solutions that are free of bias or discrimination when the data is labeled. Figure 2.1 shows the high level diagram of the reinforcement learning approach that we attempted.

#### Environment

Environment is the agent's world in which it lives and interacts in reinforcement learning. The agent can interact with the environment by taking some actions, but those actions cannot change the environment's laws or dynamics. This indicates that if humans are the agents in the earth's environments, we are bound by the planet's physics rules. Our activities can affect the environment, but we can't change the physics of our world. Here, a decision-making program is known as an agent. In the real world, we can describe an agent as a learner. When an agent does an action in the environment, the environment returns a new state, which helps the agent take

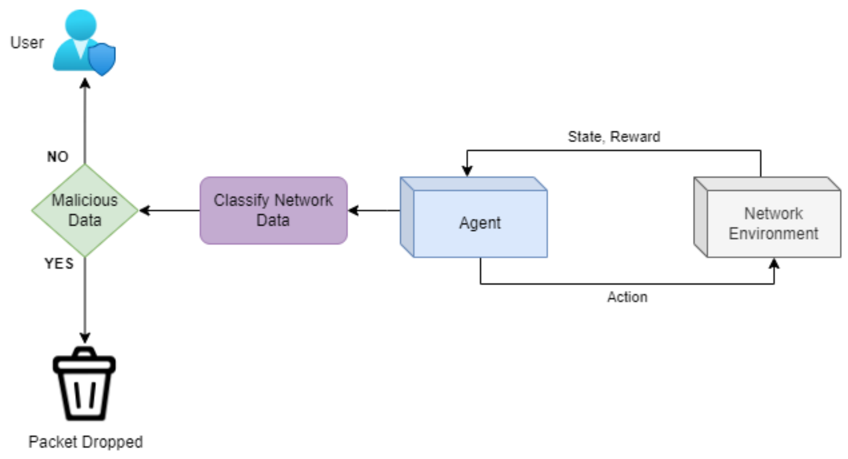


Figure 2.1: Reinforcement Learning in Network Data Classification Context

the next action. The environment also provides the agent a reward, which the agent may use to gauge the effectiveness of their action and receive feedback on whether or not their behavior was successful.

### SMOTE - Synthetic Minority Over-sampling Technique

SMOTE, as the name suggests, is a minority class oversampling technique used to increase the number of samples of a class [2]. It is very helpful in scenarios where the dataset is imbalanced. It helps us reduce the bias a model might have due to imbalanced data. The algorithm uses existing data of a class to plot a line and then uses K - Nearest Neighbor algorithm for each example to find one or more synthetic data.

### Advantage Actor Critic Algorithm

A2C is a reinforcement learning algorithm [4] that uses a combination of both value based and policy based methods. Because of policy-based mechanisms, A2C is better for continuous and stochastic situations, has faster convergence, and is more sample efficient and steady due to the value-based approach. It uses two deep neural networks, where one is called the actor network and the other is the critic network. The actor network uses the policy based method while the critic network uses value based method. The actor network is responsible for how the agent acts in the environment while the critic network outputs the quality value of actions taken by the agent to show how good the action was. One of the prominent features of A2C is that it uses the Advantage function to stabilize the model and lower the high variance of policy networks. It depicts how good the action is, as well as how much better it could be. So it basically drives the model to a stable mean value reward which eventually helps to get a better accuracy.

### Proximal Policy Optimization Algorithm

PPO [5] is another reinforcement learning algorithm that uses policy based methods. It's an on-policy gradient approach in action. Policy gradient methods are reinforcement learning approaches that use gradient descent to optimize parameterized policies in terms of expected return or long-term cumulative reward. Trust

Region Policy Optimization (TRPO) has been improved by PPO. TRPO's most major flaw is that it is computationally expensive, whereas, PPO is a simplified version of TRPO that uses a clipped surrogate objective to achieve comparable results. In other words, PPO entails collecting data through contact with the environment and employing stochastic gradient descent to optimize a surrogate objective function. Additionally, PPO makes balance between important aspects like ease of application, ease of calibration, sample complexity and so on. Furthermore, PPO addresses the issue of sampling efficiency or more specifically how to reach a certain level of accuracy with fewer simulations and less computational time. It looks for actions in the current state. This means that the agent learns from its interactions with the environment when in a certain state. The decision-making policy is updated using a minibatch of agent experiences. At each step, PPO tries to compute an update that minimizes the cost function while keeping the deviation from the preceding policy to a minimum.

### 2.2.2 Convolutional Neural Network

A deep learning network architecture that learns instantly from input is a convolutional neural network (CNN or ConvNet). CNN's design was influenced by how the human brain functions. Using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, CNN is designed to continually and smoothly learn spatial hierarchies of feature through backpropagation [7]. An identical design consists of a stack of several convolution layers, a pooling layer, followed by one or more fully connected layers. CNNs are very advantageous for recognizing objects, classes, and categories by exploring patterns. Since a feature could appear anywhere in the image, CNN is frequently used by researchers for image processing.

#### Activation Function

The activation function computes the weighted sum and then adds bias to it to decide whether or not to trigger a neuron. The activation function's objective is to add non-linearity to a neuron's outcome. The activation function has a value of 0 or 1. If the value is 1, the function will be activated; if it is 0, the opposite will occur. We are using the Softmax function as the activation function for the output layer. Softmax function assigns decimal probability to each class in a multi-class problem. Not only does it map the output from 0 to 1 range, but it also maps every outcome so that the aggregate of all of them equals 1.

$$Softmax(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}}$$

### 2.2.3 Meta Learning

Meta learning is one of the rising and exhilarating research domains in the AI field. Meta Learning basically breaks the traditional way of training a model with huge data samples. It introduces the idea of training a model on various related tasks with fewer data samples and it can use this learning for related new future tasks. This allows us to deploy machine learning techniques in domains with a very limited amount of data. Few shot learning is one such technique of meta learning.

### Support Set

The term ”support set” is used in the context of meta learning. If we take a dataset called D and randomly select some data samples from each of the classes within it without replacing them to create a different dataset, then that dataset will be regarded as a support set. An N-way K-shot support set will contain N different classes of data and each of the N classes will have K number of data points. For example, a 5-way 2-shot support set will have 5 classes of data and each class will have 2 data points for a total of 5 x 2 = 10 data points. Data from this set provides the network with support information that it will need to predict unknown data.

### Query Set

The theory of a query set is another meta learning phenomenon. Similar to how the support set was chosen, we will pick data points from dataset D at random, but using different data samples. While the support set is used to support the network, data in the query set is used to query the network. While training, the network uses the support information and tries to predict on the data from the query set and then finds the loss to update the network. While testing, data from the testing dataset go into the query set while training data is used for support.

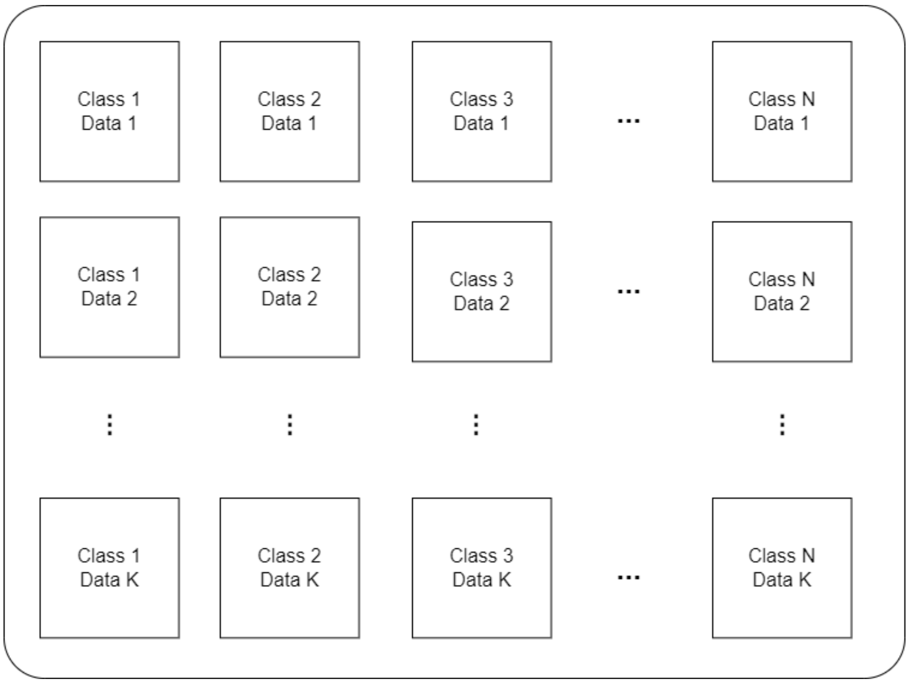


Figure 2.2: Illustrative Representation of Support and Query Set

### Few Shot Learning

Few-shot learning is a learning method where the training dataset contains limited data. It is also known as n-way k-shot learning where k denotes the number of data points of n classes. To address the identification of unknown classes, few-shot learning models have been developed to accomplish work with a constrained amount of training samples.

### Siamese Network

Siamese Network is a successor of the meta learning approach. It employs fewer data samples to address issues where there isn't enough information to train a model. It is one of the most highly sophisticated few-shot learning techniques and consists of two symmetrical neural networks with identical architectures and weights in each network. The prime objective of the Siamese network is to determine whether the two inputs that passed through the two networks are similar or not.
Example: Suppose $X_1$ and $X_2$ are two symmetric networks and $A_1$ and $A_2$ are two image inputs. We passed them through $X_1$ and $X_2$ respectively. The networks will use CNN to extract the features from the images and give embeddings for the inputs. After collecting the embeddings, it will be sent to an Energy Function that actually uses any distance function to measure the similarity between inputs. If the distance is lower than the threshold value, the inputs belong to the same class and otherwise not.

### Prototypical Network

Another successor of the meta learning technique is prototypical network. A normal network tries to learn the metric space in order to perform classification. The foundation of prototypical network is the notion that each class should have a prototypical representation, and that each query point should be classified based on how closely it resembles the class prototype. The architecture of our prototypical network is shown in figure 4.6.
Example: Suppose we have three classes $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. Every class has $n$ samples which represented as $\mathbf{X} = \{X_1, X_2, X_3, ..., X_n\}$, $\mathbf{Y} = \{Y_1, Y_2, Y_3, ..., Y_n\}$, $\mathbf{Z} = \{Z_1, Z_2, Z_3, ..., Z_n\}$. When all the data samples are sent into the network, the network will use CNN to extract the features and get the mean of embedding for each class.

$$\mathbf{X}_{prototype} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

In this way we will get the average embeddings for every class that are representative of prototype classes like $\mathbf{X}_{prototype}$, $\mathbf{Y}_{prototype}$ and $\mathbf{Z}_{prototype}$ . Now in the testing phase, for every point P, we will calculate the embedding $P_{embedding}$ and then, we compute the distance between class prototype and $P_{embedding}$. Then, we apply softmax to this distance and get the probabilities. And from the highest probability, we get the class of query point P.

### Energy Function

In networks such as the Siamese and Prototypical network, energy functions are used to find the similarity between 2 or more inputs. This allows us to determine the classification of the data we are trying to classify and also to find the loss incurred by the network. While implementing our approach, we have used Euclidean distance function as our energy function. It takes the embeddings of two inputs and finds the distance between them. The general formula of euclidean distance function is as follows

$$E(\mathbf{X_1}, \mathbf{X_2}) = ||f(\mathbf{X_1}) - f(\mathbf{X_2})||$$

where $\mathbf{X_1}$ and $\mathbf{X_2}$ are two feature vectors.

### Learning Rate

The learning rate is a hyperparameter that determines to regulate the model every time in such a way that the model weights are modified in response to the predicted error. To put it another way, learning rate refers to how quickly a neural network updates the data it has acquired. When constructing a neural network, the learning rate can be the most crucial hyperparameter. It can be difficult to choose the learning rate as because of a too small value, the training process can get stuck and on the other side, a too large value could lead to an unstable training process. A productive learning rate is high enough to train in a reasonable amount of time while yet being low enough for the network to converge on something valuable.

# Chapter 3

# Dataset

One of the core component of research based study is dataset. As we worked on cyber security domain, there is very limited amount of publicly available resources are present. Among them we have used are CSE-CIC-IDS2017 and CSE-CIC-IDS2018. At the time of our work, these are the latest set of publicly available datasets. The main aim of our model is to correctly predict whether network data is malicious or benign. The different labels in the datasets are shown in table 3.1.

## 3.1 Data Collection

### 3.1.1 CSE-CIC-IDS 2018

This dataset contains 78 features and 15 different labels. The dataset is divided into 10 files, where each file contains benign and a type of data e.g. DDoS, DoS, Bruteforce, etc. The 78 features of the dataset were extracted using CICFlowMeter. The dataset is heavily unbalanced as evident from the disproportionate distribution of data between malicious and benign data points. There are 13390249 benign network flow data points whereas all malicious network flow combined have 2746934 data points.

### 3.1.2 CSE-CIC-IDS 2017

This dataset contains 9 different files that records data from 9 different days. It contains 78 features just like the CSE-CIC-IDS2018 dataset. However, all the data labels are not the same in both datasets. In the CSE-CIC-IDS2018 dataset, they categorize the DDoS attacks where we found the generalized DDoS attacks in CSE-CIC-IDS2017 dataset. Here we also got portscan, heartbleed which were not present in the CSE-CIC-IDS2018. The unequal distribution of data between malicious and benign data points demonstrates the dataset's severe unbalance. There are 1741839 benign network flow data points whereas all malicious network flow combined have 556556 data points.

## 3.2 Data Preprocessing

Data preprocessing is a very crucial step. The model might not provide the desired outcomes because of not having clean and appropriate data. Usually a dataset

Table 3.1: Labels and Per Label Samples in IDS2017 and IDS2018

| Label | CSE-CIC-IDS17 | CSE-CIC-IDS18 |
|---|---|---|
| Benign | 1741839 | 13390249 |
| DDoS | 128025 | N/A |
| DDoS attack-HOIC | N/A | 686012 |
| DDoS attacks-LOIC-HTTP | N/A | 576191 |
| DoS attacks-Hulk | 230124 | 461912 |
| Bot | 1956 | 286191 |
| FTP-BruteForce | 7935 | 193354 |
| SSH-Bruteforce | 5897 | 187589 |
| Infilteration | 36 | 160639 |
| DoS attacks-SlowHTTPTest | 5499 | 139890 |
| DoS attacks-GoldenEye | 10293 | 41508 |
| DoS attacks-Slowloris | 5796 | 10990 |
| DDoS attack-LOIC-UDP | N/A | 1730 |
| Brute Force -Web | 1507 | 611 |
| Brute Force -XSS | 652 | 230 |
| SQL Injection | 21 | 87 |
| PortScan | 158804 | N/A |
| Heartbleed | 11 | N/A |

contains a lot of information that might be irrelevant for the model. In that case, those irrelevant data might lessen the efficiency of that model and increase the time complexity as well. Different models demand different data preprocessing. As we mentioned above, we worked on two different mechanism and they required much different preprocessing from each other. Both procedures are outlined below.

### 3.2.1 Data Processing for Reinforcement Learning

CSE-CIC-IDS2018 dataset is heavily unbalanced. Figure 3.1 depicts the degree of disproportion of the data in the merged dataset. The number of malicious network traffic activity is not very frequent and that is being depicted in the dataset. As a result, Benign is much more prominent than the other labels. The lowest number of data points for a class label is 87, which is for SQL Injection attacks. As a result, we decided to oversample the data. Before oversampling, we remove the null and infinity values from the dataset. Then, we extracted 150000 samples from the individual classes wherever possible and oversampled the data for the minority classes where the number of data is less than 150000 using SMOTE. After oversampling and balancing the dataset, the data count per label is shown in table 3.2.

Table 3.2: Data Count Per Label After Oversampling IDS2018

| Label | Count |
|---|---|
| Benign | 150000 |
| DDoS attack-HOIC | 150000 |
| DDoS attacks-LOIC-HTTP | 150000 |
| DoS attacks-Hulk | 150000 |
| Bot | 150000 |
| FTP-BruteForce | 150000 |
| SSH-Bruteforce | 150000 |
| Infilteration | 150000 |
| DoS attacks-SlowHTTPTest | 150000 |
| DoS attacks-GoldenEye | 150000 |
| DoS attacks-Slowloris | 150000 |
| DDoS attack-LOIC-UDP | 150000 |
| Brute Force -Web | 150000 |
| Brute Force -XSS | 150000 |
| SQL Injection | 150000 |

Following the oversampling of the dataset, we discovered that various features, such as Fwd Byts/b Avg, Fwd Pkts/b Avg, and others, have only one value, which is 0. As a result, we chose to drop them because they have no bearing on the performance of the model. Following that, we look for a correlation between each feature and the label. We then eliminated features with less than 10% correlation with the label. We trained models with and without those features, and discovered that the difference

Table 3.3: Label Encoding of Each Class

| Label | Encoded Label |
|---|---|
| Benign | 0 |
| Bot | 1 |
| Brute Force -Web | 2 |
| Brute Force -XSS | 3 |
| DDos attack-HOIC | 4 |
| DDos attack-LOIC-UDP | 5 |
| DDos attacks-LOIC-HTTP | 6 |
| Dos attacks-GoldenEye | 7 |
| Dos attacks-Hulk | 8 |
| Dos attacks-SlowHTTPTest | 9 |
| Dos attacks-Slowloris | 10 |
| FTP-Bruteforce | 11 |
| Infilteration | 12 |
| SQL Injection | 13 |
| SSH-Bruteforce | 14 |

was quite negligible. As a result, we decided to remove these features in order to achieve faster performance. The histogram of correlation between each feature and the labels are shown in Figure 3.1.

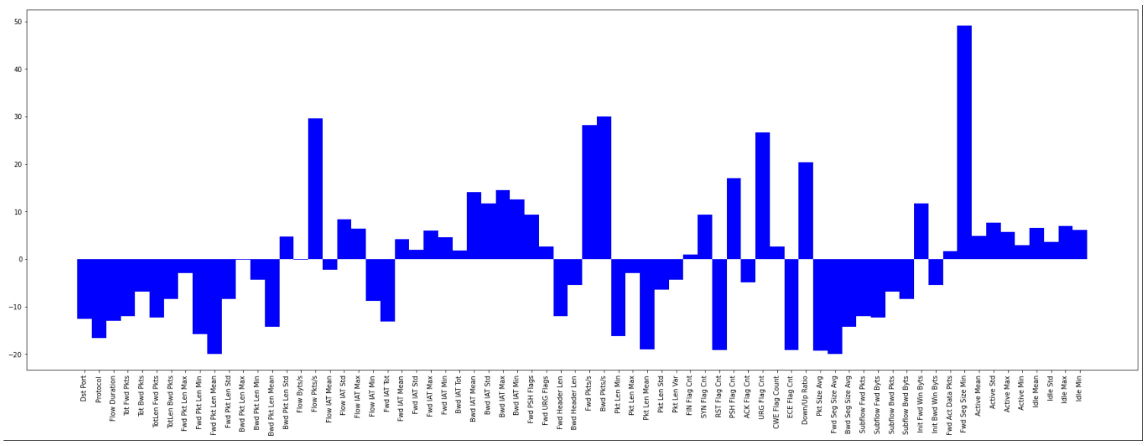
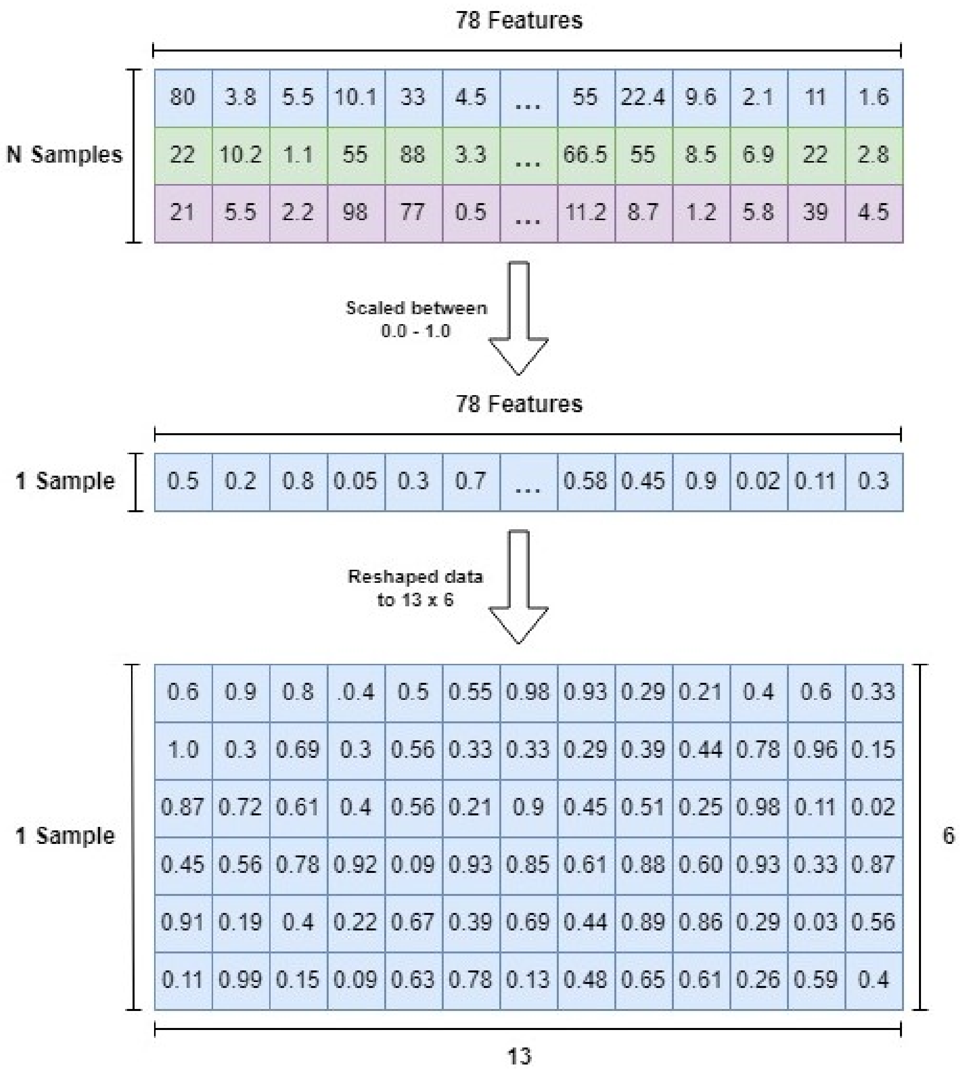


Figure 3.1: Correlation of features with the class labels

As each class label was categorical, we decided to encode the labels. The label encoding is shown in table 3.3.
We then proceeded to split the data into train, validation and test set. 70% of the data was turned into a train set, 15% into a validation set and the last 15% into a test set.

### 3.2.2 Data Processing for Hybrid Meta Learning

CNNs are renowned for their effectiveness in feature extraction from image type data. We prioritize CNN for our feature extraction task because it is excellent at extracting features, and we consider the outcomes as our features for further analysis. As a result, unlike our previous RL approach, we do not manually remove any features. So, first we remove all the null and infinity values, then scale the data between 0 and 1 and lastly, we proceed to convert each data point into image data. The 78 features of the dataset were converted to a shape of 13 x 6. As we will be using grayscale images, the final shape of each data point becomes 13 x 6 x 1. Figure 3.2 shows the creation of image data.



| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 80 | 3.8 | 5.5 | 10.1 | 33 | 4.5 | ... | 55 | 22.4 | 9.6 | 2.1 | 11 | 1.6 |
| 22 | 10.2 | 1.1 | 55 | 88 | 3.3 | ... | 66.5 | 55 | 8.5 | 6.9 | 22 | 2.8 |
| 21 | 5.5 | 2.2 | 98 | 77 | 0.5 | ... | 11.2 | 8.7 | 1.2 | 5.8 | 39 | 4.5 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.5 | 0.2 | 0.8 | 0.05 | 0.3 | 0.7 | ... | 0.58 | 0.45 | 0.9 | 0.02 | 0.11 | 0.3 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.6 | 0.9 | 0.8 | .0.4 | 0.5 | 0.55 | 0.98 | 0.93 | 0.29 | 0.21 | 0.4 | 0.6 | 0.33 |
| 1.0 | 0.3 | 0.69 | 0.3 | 0.56 | 0.33 | 0.33 | 0.29 | 0.39 | 0.44 | 0.78 | 0.96 | 0.15 |
| 0.87 | 0.72 | 0.61 | 0.4 | 0.56 | 0.21 | 0.9 | 0.45 | 0.51 | 0.25 | 0.98 | 0.11 | 0.02 |
| 0.45 | 0.56 | 0.78 | 0.92 | 0.09 | 0.93 | 0.85 | 0.61 | 0.88 | 0.60 | 0.93 | 0.33 | 0.87 |
| 0.91 | 0.19 | 0.4 | 0.22 | 0.67 | 0.39 | 0.69 | 0.44 | 0.89 | 0.86 | 0.29 | 0.03 | 0.56 |
| 0.11 | 0.99 | 0.15 | 0.09 | 0.63 | 0.78 | 0.13 | 0.48 | 0.65 | 0.61 | 0.26 | 0.59 | 0.4 |

Figure 3.2: Conversion of 1D data from CSE-CIC-IDS2018 to 2D image data

# Chapter 4

# Methodology

In this study, we worked on both reinforcement learning and hybrid meta deep learning. Following section provides a comprehensive study on both of the methodologies.

## 4.1 Reinforcement Learning Approach

In reinforcement learning, different environments have different lengths of episodes, where when an agent takes an action, the environment changes accordingly. Usually, the actions taken in one episode affects the states in that episode, but does not affect the next episode. However, the task we are trying to perform is classification, and therefore, the action that the agent performs does not change the state. But, the agent will infer that the action it is taking is causing the change. As a result, each episode in our environment only provides one data sample to the agent. Figure 4.1 demostrates the overall workflow of our RL approach.

The dataset contains 15 different classes, which the agent has to predict given a data point, $s_i$. We initialize a data counter to keep track of which data is being passed to the agent at any time step. The reset function is called to go back to the initial condition of the environment. It is important to call this function at the end of every episode. As each episode is equal to one data sample from the dataset, we use this function to iterate through the dataset. This function always increments the data counter and returns the next data sample as a state and if the data counter variable is greater than the number of samples in the dataset, the dataset is shuffled first. It is also saving the correct action, $c_t$, that needs to be taken for the state, $s_t$, being returned.

The environment of our agent can be separated into 3 major components. The state space of the environment which shows all the different states the environment can be in, the action space which is all the actions the agent can take in the environment, and the reward function which is used by the environment to calculate the reward for the action performed by the agent on a specific state. They are defined as:

- *State space*: The state space of our environment are all the network data features of the dataset except the label. The state space is

  $$S(t) = \{s_0, s_1, s_2, ..., s_n\}$$

  where each $\mathrm{s_i}$ is a vector of shape (1, 70) values corresponding to the 70 features of the dataset.

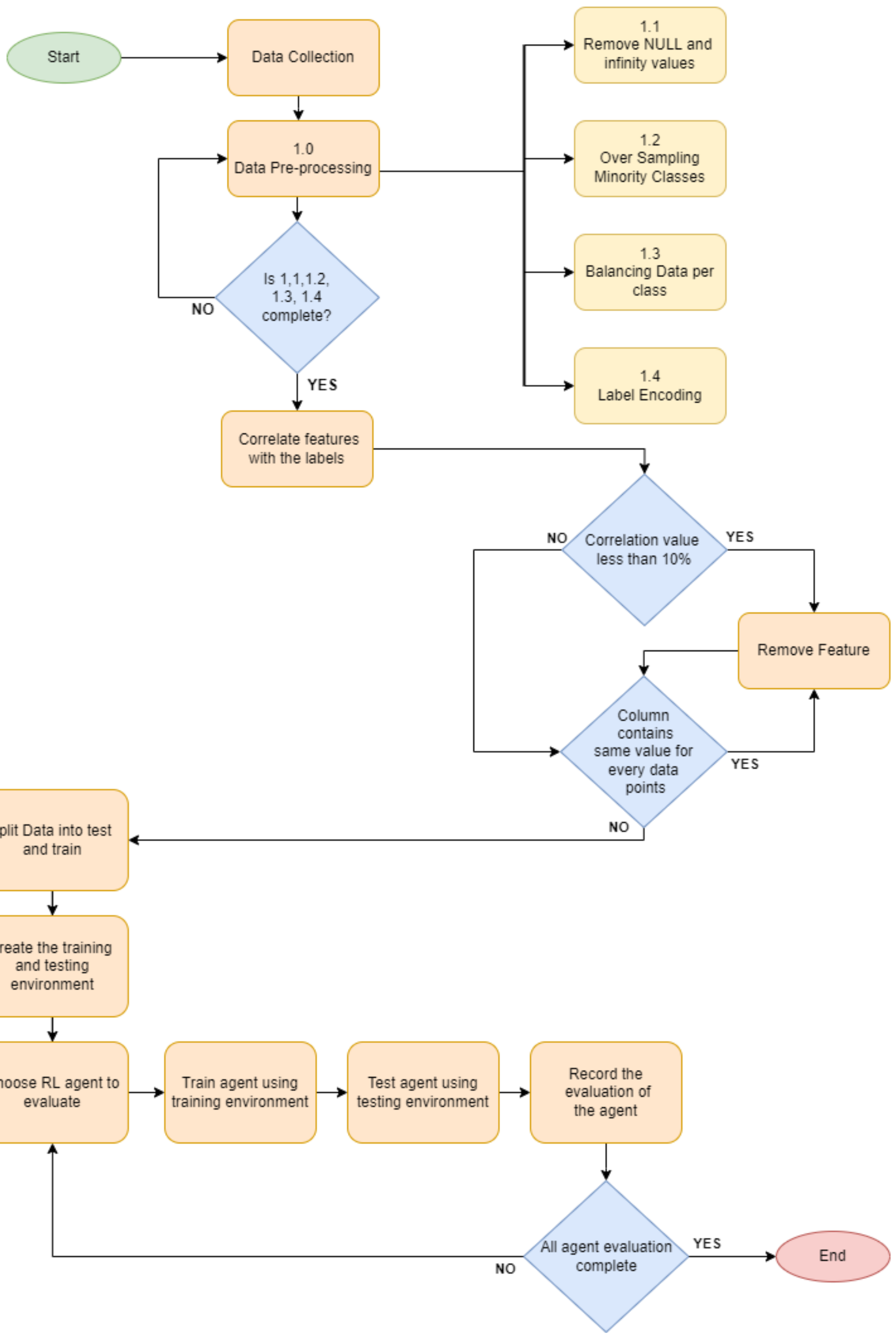

Start
Data Collection
1.0
Data Pre-processing
Is 1,1,1.2, 1.3, 1.4 complete?
NO
YES
1.1
Remove NULL and infinity values
1.2
Over Sampling Minority Classes
1.3
Balancing Data per class
1.4
Label Encoding
Correlate features with the labels
Correlation value less than 10%
NO
YES
Remove Feature
Column contains same value for every data points
YES
NO
Split Data into test and train
Create the training and testing environment
Choose RL agent to evaluate
Train agent using training environment
Test agent using testing environment
Record the evaluation of the agent
All agent evaluation complete
NO
YES
End

Figure 4.1: Top Level Overview of RL Framework.

- *Action space*: The action space of our environment are the different labels of the data. There are 15 distinct labels and therefore the action space is

$$A = \{a_0, a_1, ..., a_{14}\}$$

  where each $a_i \in [0, 14] \subset \mathbb{Z}$

- *Reward Function*: The task we are performing is classification. Therefore, the agent should be rewarded for correctly labeling data. Thus, the reward function is defined as:

$$r(c_t, a_t) = \begin{cases} 1 & a_t = c_t \\ 0 & a_t \neq c_t \end{cases}$$

  Where $c_t$ is the correct action (or label) for current state and $a_t$ is the action predicted by the agent. The objective of the agent can be expressed as:

$$maximize \;\; R = \sum_{t=0}^{n} r(c_t, a_t)$$

Agent takes the current state from the reset function. Then, the agent predicts the action, $a_t$, and uses the step function to calculate the reward for that action. To illustrate, the step function takes action as an input and calculates the reward by using the reward function. The step function then returns the current state, reward, and done at the end. Unlike other step functions, it does not return the next state as the episode ends after each step function call and reset function will provide the next state. The agents that we use in our approach are A2C and PPO. The algorithms 1 and 2 demonstrate their working process.

**Algorithm 1** Advantage Actor Critic Algorithm

```
Input: Set parameters $\sigma$ and $s$ to their default values
  loop
      Sample $a \sim r_\sigma$
      Take action $a$, receive a reward, $r_t$ and proceed to state $s'$
      TD error Calculation:
        $\Delta_{t_i} = r(s_t, a_t) + aQ(s_{t+1}) - Q(s_t)$
      Actor parameters should be updated as follows:
        $\sigma \leftarrow \sigma + \beta(\frac{1}{M}\sum_{j=1}^{M}\left[\sum_{t=0}^{T} a^t \Delta_\sigma \log r_\sigma(a_{j,t}|s_{j,t})(r(s_t, a_t) + aQ(s_{t+1} - Q(s_t)))\right])$
      Update $Q(s_t)$ using target $r(s_t, a_t) + aQ(s_{t+1})$
  end loop
The process will continue until the expected state $s$ is achieved
```

After training the agents for 2 epochs, the average reward per episode is shown in figure 4.2.
It is clear from the graph that both agents move to the mean reward range of 0.6 to 0.8 pretty rapidly. However, both agents are quite consistent in this region afterwards.

**Algorithm 2** Proximal Policy Optimization Algorithm

```
Input: initialize policy parameters $\Phi_0$, clipping threshold $\delta$
  for i = 0, 1, 2, ... do
      By running the policy $\sigma_i = \sigma(\phi_i)$, collect a set of trajectories $T_k$
      Using any advantage estimation algorithm, calculate the advantage $\hat{A}_t^{\sigma_i}$
      Updated calculation policy
          $\Phi_{i+1} = \arg\max_{\Phi} L_{\Phi_k}^{CLIP}(\Phi)$
      Performing I stages of mini batch SGD (using Adam)
          $L_{\Phi_k}^{CLIP}(\Phi) = E_{\tau\sim\sigma_i}\left[\sum_{t=0}^{T}[min(p_t)(\Phi)\hat{A}_t^{\sigma_i}, clip(p_t(\Phi), 1-\delta, 1+\delta)\hat{A}_t^{\sigma_i})]\right]$
  end for
```

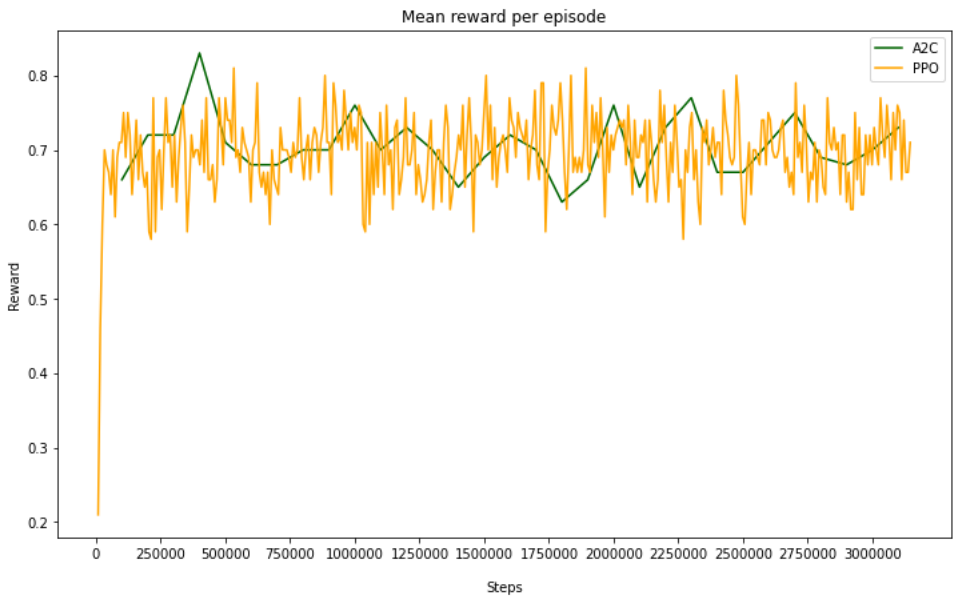


Figure 4.2: Average reward for per episode

## 4.2 Hybrid Meta Deep Learning Approach

The block diagram of our proposed model is illustrated in figure 4.3. It is a hybrid combination of two few shot learning techniques, Siamese Network and Prototypical Network. The following sections describe the architecture of each individual network.

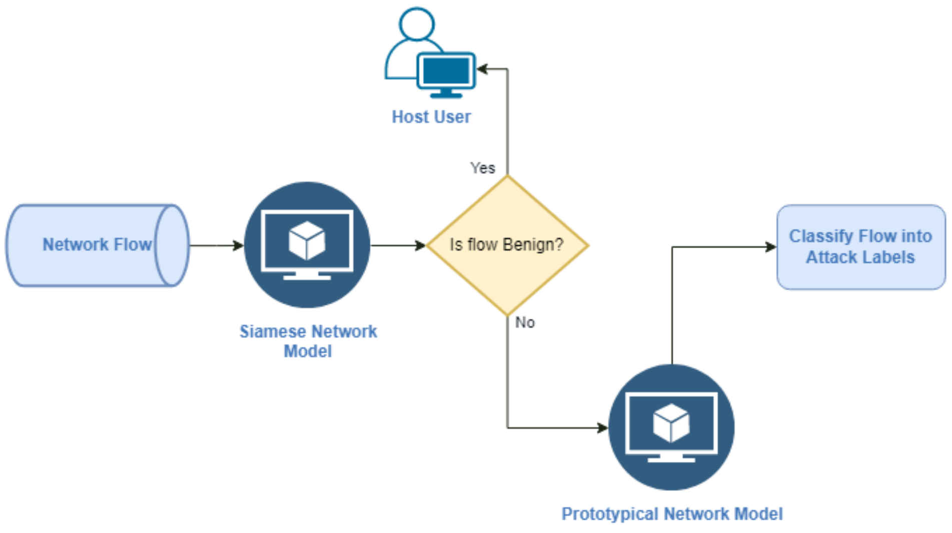


Figure 4.3: High Level Diagram of our Hybrid Meta Learning Approach.

The Siamese network is being used for binary classification to classify the data into benign and malicious. The feature extractor of the Siamese network contains two convolutional and max pool layers each with 64 filters. The kernel size of the convolutional layers that give us the best outcome is 4 x 4 with a max pooling size of 2 x 2. The architecture of our Siamese network is displayed in figure 4.5 for better understanding. We are using euclidean distance as the energy function which is defined as:

$$D(\mathbf{X_1}, \mathbf{X_2}) = ||f(\mathbf{X_1}) - f(\mathbf{X_2})||$$

where $\mathbf{X_1}$ and $\mathbf{X_2}$ are data points and $f(\mathbf{X_1})$ and $f(\mathbf{X_2})$ are the embeddings of the data points.
For the loss function we are using contrastive loss which is defined as:

$$\text{Contrastive Loss} = Y_{\text{true}} \times D^2 + (1 - Y_{\text{true}}) \times \max(1 - D, 0)^2$$

The algorithm for the training and testing of the Siamese network is shown in algorithm 3.
The Prototypical network is being used for multiclass classification to determine which attack type the data falls into. Its feature extractor also contains 2 convolutional and max pool layers with 64 filters in each layer. Kernel size for the convolutional layers that give us the best outcome is 4 x 4 with a max pooling size of 2 x 2. Prototypical network architecture is illustrated in figure 4.6 for a visual representation. Euclidean distance is again being used as the energy function and negative log probability from the softmax layer is being used for the loss calculation.

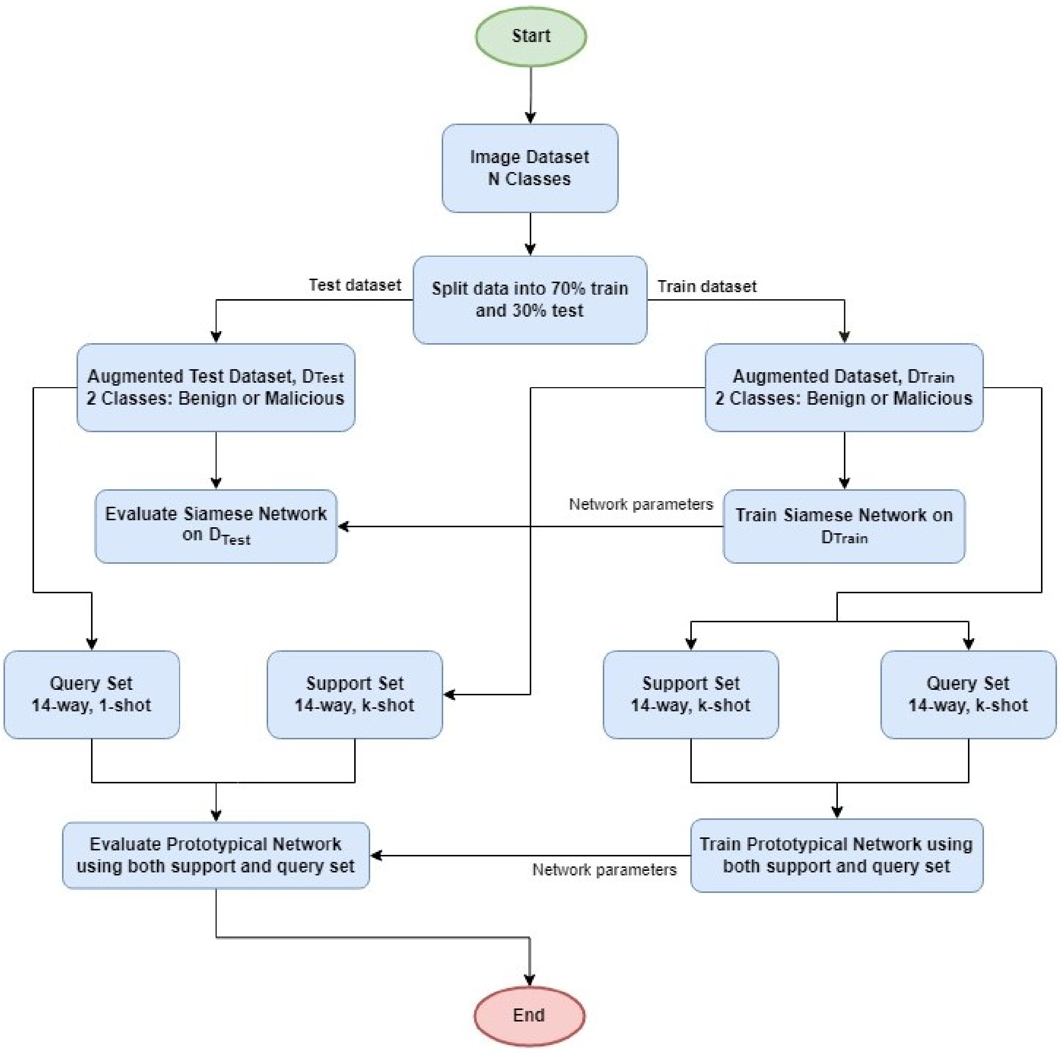

Start
Image Dataset
N Classes
Split data into 70% train and 30% test
Test dataset
Train dataset
Augmented Test Dataset, DTest
2 Classes: Benign or Malicious
Augmented Dataset, DTrain
2 Classes: Benign or Malicious
Evaluate Siamese Network on DTest
Network parameters
Train Siamese Network on DTrain
Query Set
14-way, 1-shot
Support Set
14-way, k-shot
Support Set
14-way, k-shot
Query Set
14-way, k-shot
Evaluate Prototypical Network using both support and query set
Network parameters
Train Prototypical Network using both support and query set
End

Figure 4.4: Top Level Overview of the Proposed Framework.

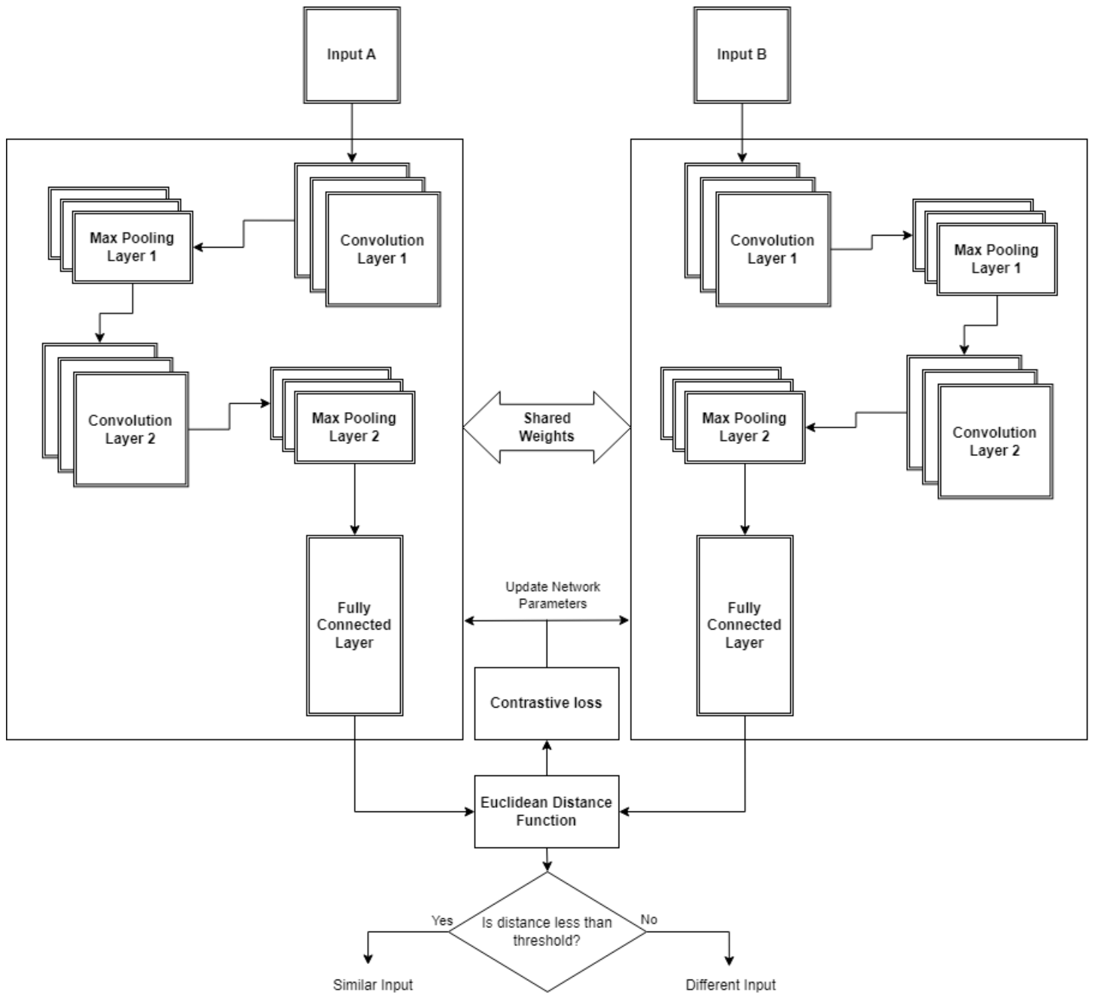


Figure 4.5: Block Diagram of the Siamese Network Architecture of our Proposed Approach

**Algorithm 3** Malicious Data Identification using Siamese Network

```
Input: Network Flow Images
Output: Malicious Data Identification, Malicious Data
$D_{Train} \leftarrow \{\mathrm{Im}_{Ai}, \mathrm{Im}_{Bj}\}$ where i and j are random indexes for the training set
$D_{Test} \leftarrow \{\mathrm{Im}_{Ai}, \mathrm{Im}_{Bj}\}$ where $\mathrm{Im}_{Ai}$ is a random benign sample from training set
and j is a random index for testing set
epoch $\leftarrow$ 0
threshold $\leftarrow$ 0.5
  while epoch $\leq$ max_epoch do
      for data in $\mathrm{D}_{Train}$ do
          Embeddings $\leftarrow f_1(\mathrm{data}[0], \mathrm{data}[1])$
          Distance $\leftarrow D$(Embeddings)
          Loss $\leftarrow$ Contrastive Loss(Distance, $\mathrm{Y}_{True}$)
          Update Network parameters using Loss
      end for
  end while
  for data in $\mathrm{D}_{Test}$ do
      Distance $\leftarrow$ Predict(data)
      if Distance $\leq$ threshold then
          Classify data as Benign
      else
          Classify data as Malicious
      end if
      if $\mathrm{Y}_{true}$ is Malicious then
          $\mathrm{D}_{Malicious} \leftarrow \mathrm{D}_{Malicious}$.append(data[1])
      end if
  end for
```

The softmax probability is calculated using:

$$P(Y_{\text{true}} = Y_{\text{pred}} \,|\, \mathbf{X}) = \frac{e^{-D(f(\mathbf{X}),\mathbf{X}_{\text{prototype}})}}{\sum_1^n e^{-D(f(\mathbf{X}),\mathbf{X}_{\text{prototype}})}}$$

where $\mathbf{X}$ is the query, $f(\mathbf{X})$ is the embedding of the query and $D$ is the euclidean distance function. Then using $P$, the loss is calculated as follows:

$$\text{Loss} = -\log[P(Y_{\text{true}} = Y_{\text{pred}} \,|\, \mathbf{X})]$$

The algorithm for the training and testing of the Prototypical network is shown in algorithm 4.

**Algorithm 4** Malicious Data Classification using Prototypical Network

```
Input: Network Flow Images, $D_{Malicious}$
Output: Malicious Data Classification
$Training\ Data$ ← Network Flow images without Benign
epoch ← 0
epoch_size ← $max\_size$
  while epoch ≤ max_epoch do
      for iter = 0; iter < epoch_size; iter++ do
          $Support_{Train}$, $Query_{Train}$ ← Get_Support_Query(training_data, n_way, k_shot)
          Prototype, Embeddings ← $f_2(Support_{Train}, Query_{Train})$
          Distances ← $D$(Prototype, Embeddings)
          $P(Y = Y_{pred}|X)$ = Softmax(Distances)
          Loss = $-\log(P)$
          Update Network parameters using Loss
      end for
  end while
  max_iter = $D_{Malicious}$.length
  for iter = 0; iter < max_iter; iter++ do
      $Support_{Test}$, $Query_{Test}$ ← Get_Test_Support_Query($D_{Malicious}$, training_data,
  n_way, k_shot)
      Prototype, Embeddings ← $f_2(Support_{Test}, Query_{Test})$
      Distances ← $D$(Prototype, Embeddings)
      $P(Y = Y_{pred}|X)$ = Softmax(Distances)
      Prediction = $max$(P)
  end for
```

The training and testing method of the above architecture is described in the following sections. Figure 4.4 depicts the top level overview.
Data is first passed to the Siamese network. The training image dataset is converted into the augmented dataset $D_{Train}$ where each data point contains two images. If both images are benign or malicious, then the data point is labeled as 1 and if the images are different from each other it is labeled as 0. This dataset is then used to train the Siamese network so that it can optimize its parameters for a similarity function. Next, the augmented test dataset $D_{Test}$ is created where each data point contains one random benign image from the training set and another random image from the testing set. This creates a dataset where each data contains a benign image

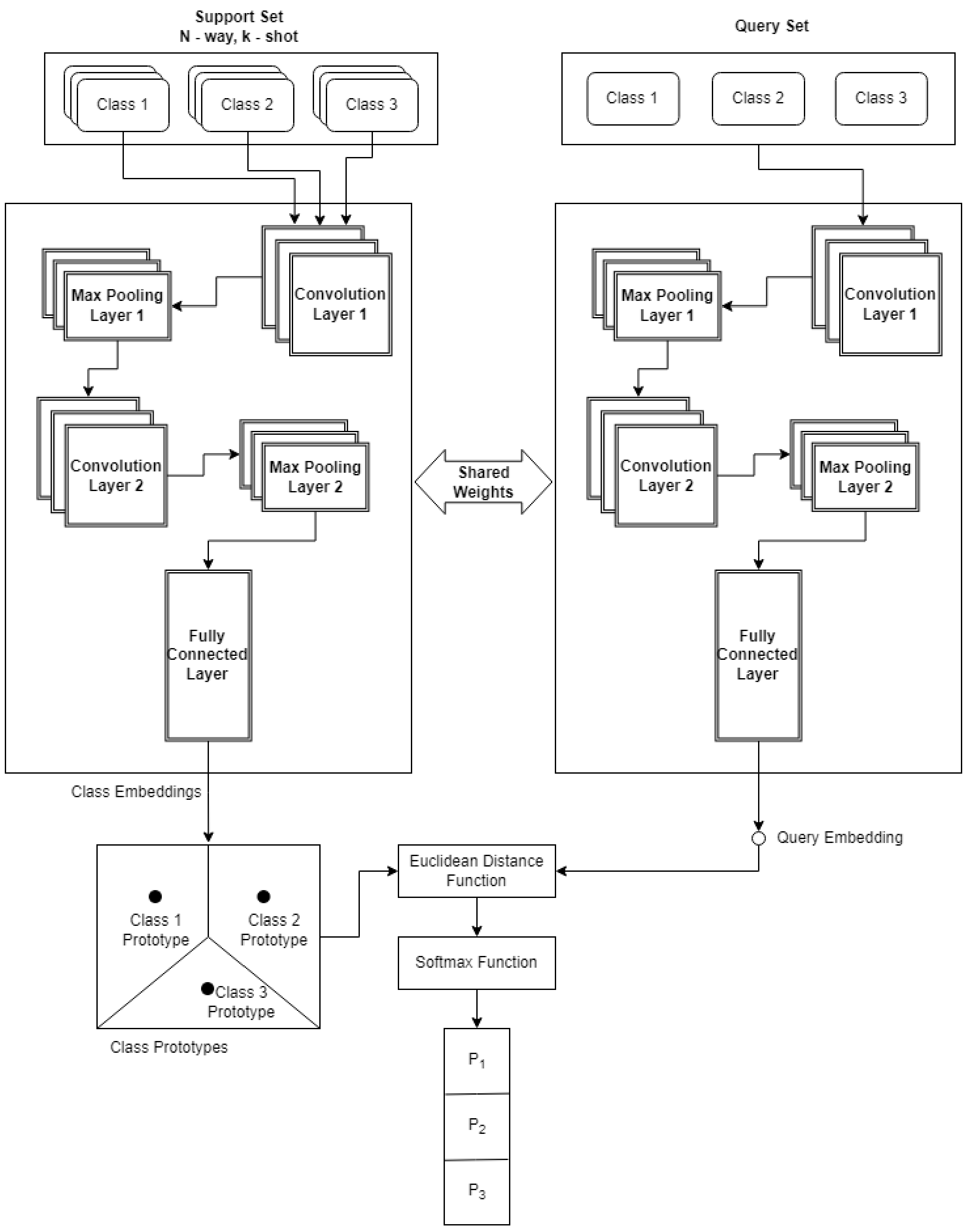


Figure 4.6: Block Diagram of the Prototypical Network Architecture of our Proposed Approach

]

from the train set and another image the model has not seen before. Labeling of data is done the same way as before. The model is then evaluated on $D_{Test}$.
The data points that are classified into malicious data are then used to train and evaluate the Prototypical network. The training malicious data images are first converted into 14-way k-shot support and query sets. Both these sets are then passed to the model for training to optimize the feature extractor. Afterwards, we create another 14-way k-shot support set using images from the training data and a 14-way 1-shot query set using images from the testing data. They are then used to evaluate the performance of the trained Prototypical network.

# Chapter 5

# Results and Discussion

## 5.1 Evaluation Metrics

Performance of our proposed framework is being evaluated by using some of the well known indicators such as accuracy, precision, recall, f1 score, specificity and ROC curve. The following indicators are calculated using the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. An outcome where the model properly predicts the positive class is referred to as a true positive and true negative results are those for which the model correctly takes the negative class into account. A false positive is a result when the model forecasts the positive class inaccurately. A false negative is a result where the model forecasts the negative class inaccurately.
Accuracy depicts the measurement of a classification system's overall efficiency. The calculation is as follows:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Among all positives, how many are correctly classified as positive is referred to as precision. The calculation is as follows:

$$Precision = \frac{TP}{(TP + FP)}$$

The ratio of accurately classified positives to actual positives is known as recall or sensitivity.The calculation is as follows:

$$Recall = \frac{TP}{(FN + TP)}$$

The harmonic mean of accuracy and recall is used to create the f1 score, which serves as an indicator of how successfully the model classifies data. Compared to the standard accuracy metric, the F1 score is seen to offer a better representation of the classifier's performance. Its value goes from 0 to 1, with 0 representing the lowest possible score and 1 representing the highest possible score. The calculation is as follows:

$$F1\ score = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$$

The ratio of correctly identified negative data to actual negative data is referred to as specificity. The calculation is as follows:

$$Specificity = \frac{TN}{(FP + TN)}$$

True Positive Rate or TPR provides the proportion of accurate forecasts in predictions of the positive class. Recall is another name for it.The calculation is as follows:

$$TPR = \frac{TP}{(TP + FN)}$$

The false positive rate (FPR) is an indicator of a test's accuracy. it provides the percentage of wrong predictions in the positive class or, in another way, it is the probability that a false alarm would be triggered. The calculation is as follows:

$$FPR = \frac{FP}{(FP + TN)}$$

ROC curves are a crucial parameter for evaluating the effectiveness of classifiers [1]. They are constructed by plotting the independent true positive rate (TPR) and false positive rate (FPR) rates. The whole test sample's TPR and FPR values are obtained at various threshold settings H in the [0-1] interval in order to plot the ROC curve. A better performance is shown by classifiers that provide curves that are closer to the top-left corner.

The area under the ROC curve, also known as the AUC score, is a scalar value that can be used to assess how well the decision model performs overall in terms of classification. A classifier's performance is better as the value gets closer to 1.0, whereas one with a value closer to 0.5 is on par with guessing labels at random.

## 5.2 Performance Evaluation of Reinforcement Learning based Approach

To implement reinforcement learning, we used A2C and PPO. In this section, using these algorithms, we will provide the results and evaluation.

The problem our agent is trying to solve is a multiclass classification problem. Therefore the evaluation metrics have to be calculated on a per class basis. We have first created a confusion matrix using the agent's prediction on the test environment. The confusion matrix for A2C and PPO is shown in figure 5.1. It shows the true labels against the predicted labels. Using the confusion matrix, we have calculated the TP, TN, FP and FN, which were then used to calculate the evaluation metrics. The evaluation of A2C and PPO are shown in table 5.1 and table 5.2 respectively.

From tables 5.1 and 5.2 it can be observed that both A2C and PPO have a good performance on correctly classifying Bot, DDOS attack-HOIC, DDOS attack-LOIC-UDP, and DOS attacks-Slowloris. Moreover, PPO seems to slightly outperform A2C

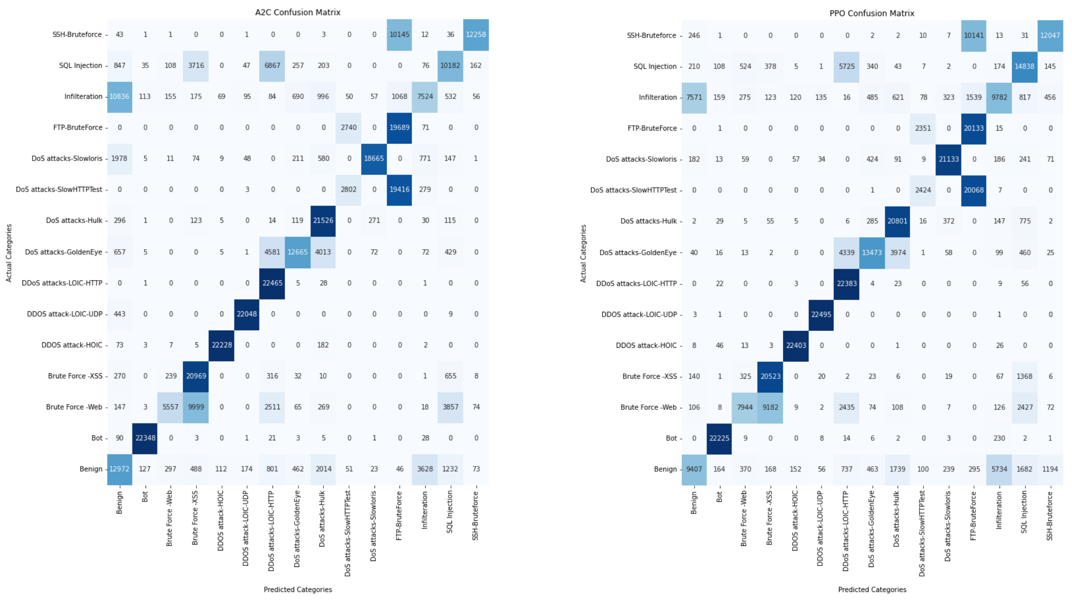


Figure 5.1: Confusion matrix of A2C (left) and PPO (right)

Table 5.1: A2C evaluation on test environment

| **Label** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|
| Benign | 0.9252 | 0.4521326684 | 0.5761777778 | 0.5066734411 |
| Bot | 0.9986222222 | 0.9860590285 | 0.9933777778 | 0.989704873 |
| Brute Force -Web | 0.9473274074 | 0.8700830591 | 0.2467555556 | 0.3844742218 |
| Brute Force -XSS | 0.9521303704 | 0.5890360972 | 0.9326666667 | 0.7220520937 |
| DDOS attack-HOIC | 0.9986222222 | 0.9909540573 | 0.9883555556 | 0.9896531007 |
| DDOS attack-LOIC-UDP | 0.9975792593 | 0.984319857 | 0.9792888889 | 0.981797928 |
| DDoS attacks-LOIC-HTTP | 0.9549362963 | 0.5968388896 | 0.9985777778 | 0.7471277745 |
| DoS attacks-GoldenEye | 0.9655288889 | 0.8748447634 | 0.5635555556 | 0.6855165703 |
| DoS attacks-Hulk | 0.9725837037 | 0.7222576424 | 0.9566222222 | 0.8230817766 |
| DoS attacks-SlowHTTPTest | 0.9332414815 | 0.4972493345 | 0.1245333333 | 0.1991825129 |
| DoS attacks-Slowloris | 0.9873659259 | 0.9774821952 | 0.8296 | 0.8974901433 |
| FTP-BruteForce | 0.9007525926 | 0.3908520607 | 0.8750222222 | 0.540344714 |
| Infilteration | 0.9408177778 | 0.601169497 | 0.3335555556 | 0.4290532815 |
| SQL Injection | 0.9426992593 | 0.5921843103 | 0.4512444444 | 0.5121957372 |
| SSH-Bruteforce | 0.9684948148 | 0.9688660608 | 0.5449333333 | 0.6975394681 |

Table 5.2: PPO evaluation on test environment

| Label | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Benign | 0.936237037 | 0.5274141211 | 0.4189777778 | 0.4669837024 |
| Bot | 0.9975466667 | 0.9758475321 | 0.9876444444 | 0.9817105496 |
| Brute Force -Web | 0.9521125926 | 0.8290074751 | 0.3548888889 | 0.4970123242 |
| Brute Force -XSS | 0.9648118519 | 0.6745186937 | 0.9124888889 | 0.7756620953 |
| DDOS attack-HOIC | 0.9987051852 | 0.9849650503 | 0.9957777778 | 0.9903419011 |
| DDOS attack-LOIC-UDP | 0.9992385185 | 0.9889646076 | 0.9997333333 | 0.9943198143 |
| DDoS attacks-LOIC-HTTP | 0.96032 | 0.6277061133 | 0.9948444444 | 0.769738652 |
| DoS attacks-GoldenEye | 0.9670903704 | 0.8663579651 | 0.5987111111 | 0.7080869405 |
| DoS attacks-Hulk | 0.9753659259 | 0.7587552897 | 0.9244 | 0.8334268312 |
| DoS attacks-SlowHTTPTest | 0.9328681481 | 0.4842779892 | 0.1074666667 | 0.1758993198 |
| DoS attacks-Slowloris | 0.9929303704 | 0.9538767037 | 0.9393777778 | 0.9465717229 |
| FTP-BruteForce | 0.8980296296 | 0.3858344672 | 0.8948444444 | 0.5391856246 |
| Infilteration | 0.9421925926 | 0.5905073253 | 0.4335111111 | 0.4999743708 |
| SQL Injection | 0.9539348148 | 0.6528602207 | 0.6599111111 | 0.6563667308 |
| SSH-Bruteforce | 0.9632237037 | 0.8583404376 | 0.5369777778 | 0.6606517935 |

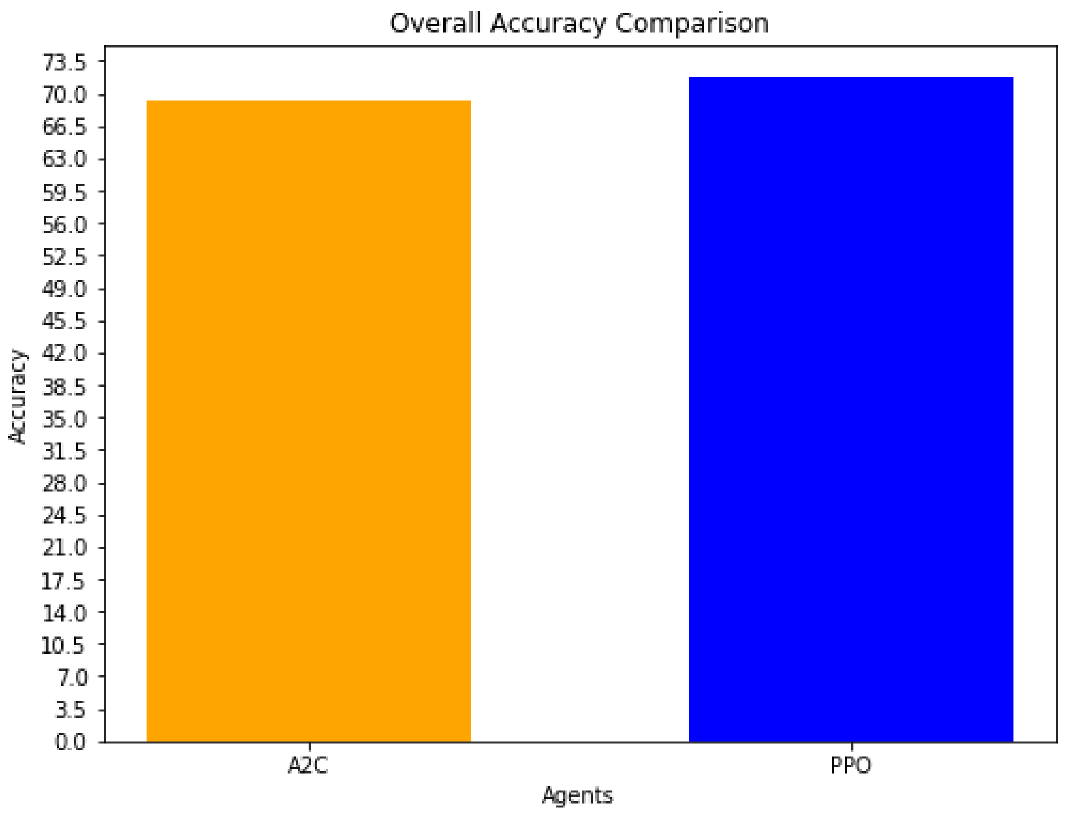


Figure 5.2: Overall Accuracy Comparison of A2C and PPO

in these classes and that is also reflected in the overall accuracy of the agents in figure 5.2.

However, this performance is achieved by training the model with 1575000. Here, every class has 150000 data samples. In the original dataset, the data samples for every label was unbalanced. So we oversampled the data to avoid any kind of biasness. But, even with such a huge number of training samples, we could only achieve an overall accuracy of around 70% for both the algorithms.Moreover, oversapmpled data is not always a very accurate depiction of real world data.
F-1 score is a better metric to consider when dealing with imbalanced datasets. Here, even though the agents have a high F-1 score for few labels such as Bot, DDOS HOIC, DDOS LOIC-UDP, for most labels the scores are quite low. This shows that the agents are not able to get high true positive values which also explains the low overall accuracy.

## 5.3 Performance Evaluation of Hybrid Meta Deep Learning based Approach

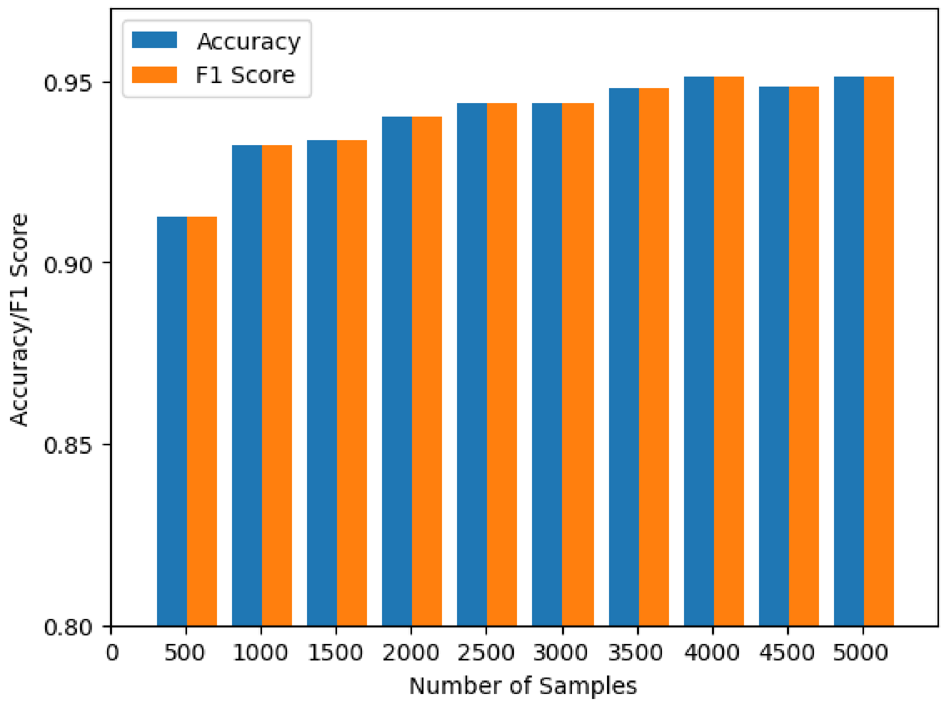
In this section, we mainly focus on the outcomes from our the proposed meta learning model. Here the main purpose of our model is to at first detect if the network packets are malicious or not and classify it if it is deemed malicious. To achieve this, we used the CSE-CIC-IDS 2018 dataset and we also used CSE-CIC-IDS 2017 to illustrate the fact that the outcomes of our model are not dataset specific. Moreover, we also compare our methods to some of the existing works in this domain.
In our proposed architecture, we randomly selected 3000 samples from our training dataset while keeping the number of samples per class balanced. Then we train our models for multiple epochs in this small sampled dataset and test it on the entire test dataset.
For binary classification using Siamese network, we found that the overall accuracy of our model is 94.36%. The f1 score, precision and recall of our model are 93.93%, 94.67% and 94.36% respectively. Given that CSE-CIC-IDS 2018 is a very unbalanced dataset, the high f1 score depicts that the unbalanced nature of the dataset is not affecting our model's performance. While using CSE-CIC-IDS 2017, we got an overall accuracy of around 94.13%. The f1 score, precision and recall of our model with the 2017 dataset are 96.11%, 94.19% and 94.13% respectively. This shows that the performance of our is consistent irrespective of the dataset. The detailed view of all of these outcomes are displayed in table 5.3.

Table 5.3: Binary Classification Results

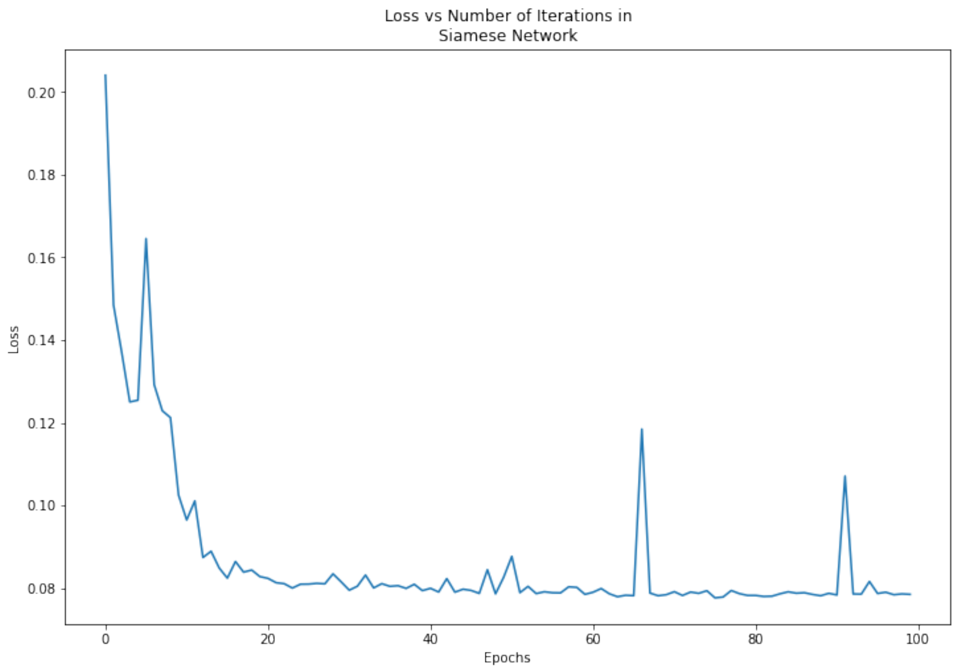
| Dataset | Precision | Recall | F1 Score | Accuracy |
|---|---|---|---|---|
| CSE-CIC-IDS2017 | 94.19 | 94.13 | 96.11 | 94.13 |
| CSE-CIC-IDS2018 | 94.67 | 94.36 | 93.93 | 94.36 |

Moreover, figure 5.3a illustrates the accuracy and f1 score variation of the outcomes for the different number of samples used for the training of the Siamese network. It always follows an upwards trend and the f1 scores are very close to the accuracy values. It reaches its peak point when the model is trained at around 4000 data samples. However, to be consistent with the hybrid model, we used 3000 data samples to train our Siamese network just as the prototypical network. Using a Siamese network trained over 3000 data samples, we got approximately 94.36% accuracy and 93.93% f1 score. Additionally, a graph is added in figure 5.3b to demostrate the validation loss of the Siamese network over 100 epochs. As seen from the graph, the loss seems to stabilize at around 80 to 90 epochs with minor changes even though there is one anomaly.

Following the Siamese network's results, we incorporated the malicious data into our prototypical network for multiclass categorization. In this experiment, we observed that the accuracy of our model is approximately 90.64% for 14 different labeled malicious network data using CSE-CIC-IDS2018. Moreover, for the IDS2018 dataset, the f1 score, precision and recall score of our model are 91%, 91.66% and 90.64% respectively. Among the 14 labels, DDoS LOIC HTTP has the highest f1 score, precision and recall. We got around 99.97% f1 score, 100% recall and 99.90% precision for DDoS LOIC HTTP. On the other hand, for SQL Injection, we got 2.49% f1 score, 53.85% recall and 1.27% precision. The huge gap of evaluation metric between DDoS LOIC HTTP and SQL Injection exists because the number of SQL Injection samples in the IDS2018 dataset is extremely low compared to others. Therefore, the network had fewer samples to create the prototype for SQL Injection attacks. However, our model works quite well for the labeled data where the number of samples is around 200. For example, using 214 DDoS LOIC UDP data samples, we got around 94.49% precision, 98.85% recall and 96.62% f1 score.
For clear visual representation we added a bar chart shown in figure 5.4a that shows the accuracy and f1 score variation of the prototypical model using CSE-CIC-IDS2018 dataset. It demonstrates the variation of the outcomes for using different number of samples during training. It is seen that both the accuracy and f1 score are following an upwards trend. However, the model gets the highest result when it is trained over 3000 data samples. Using 3000 data samples, the model accuracy and f1 score are approximately 90% and 91% respectively. Along with that, another graph is shown in figure 5.4b to illustrate the validation loss of the prototypical network over 5000 iterations. As seen from the graph, the loss seems to stabilize at around 3000 iterations. It's a downward trend graph and the loss is minimum when the model is trained over 3000 iterations.
Furthermore, for the robustness of our proposed architecture, we train our model using CSE-CIC-IDS2017. Using the IDS2017 dataset, we receive approximately 95.68% accuracy for 14 different labels. The average f1 score, precision and recall of our model are 96.1%, 96.5% and 95.68% respectively. Among all the labels, we got the highest f1 score, precision and recall from Heartbleed. However, there were only 11 samples in the total dataset and only 4 samples in the testing dataset. So, we do not believe it correctly portrays the performance of the model and are therefore considering PortScan as the best performing label. For PortScan we got around 99.90% precision, 99.08% recall and 99.49% f1 score. On the contrary, because of the very small amount of data, we got approximately 1.27% precision, 100% recall and 2.51% f1 score for SQL Injection data. Table 5.5 shows the metrics for the individual labels of both CSE-CIC-IDS2018 and CSE-CIC-IDS2017.
We know, hyper parameters are important to choose as appropriate parameters help

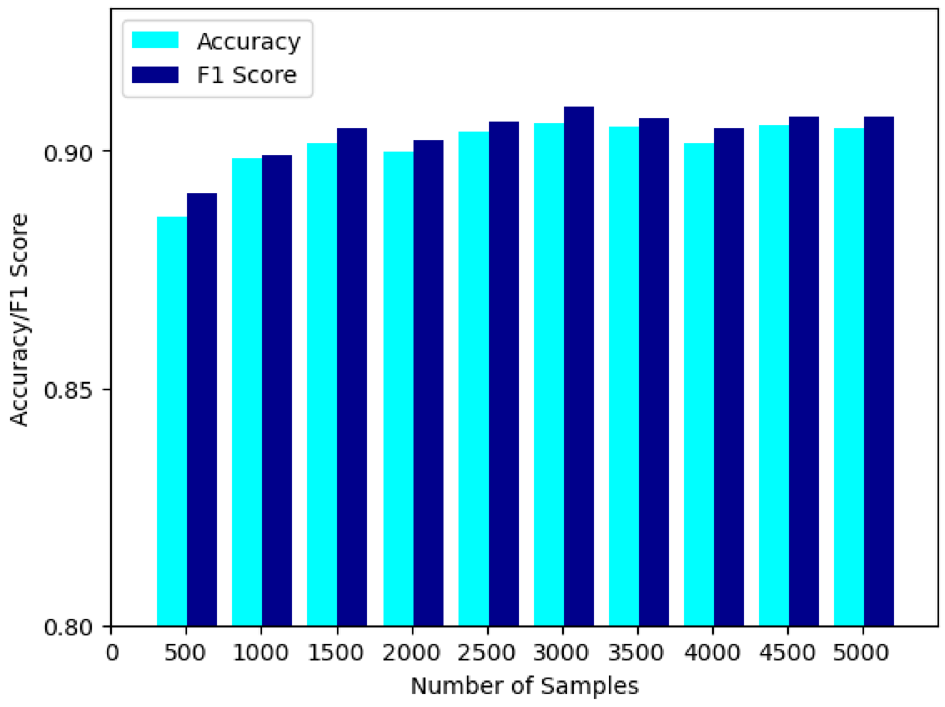


(a) Accuracy and F1 vs No. of Samples of Siamese Network

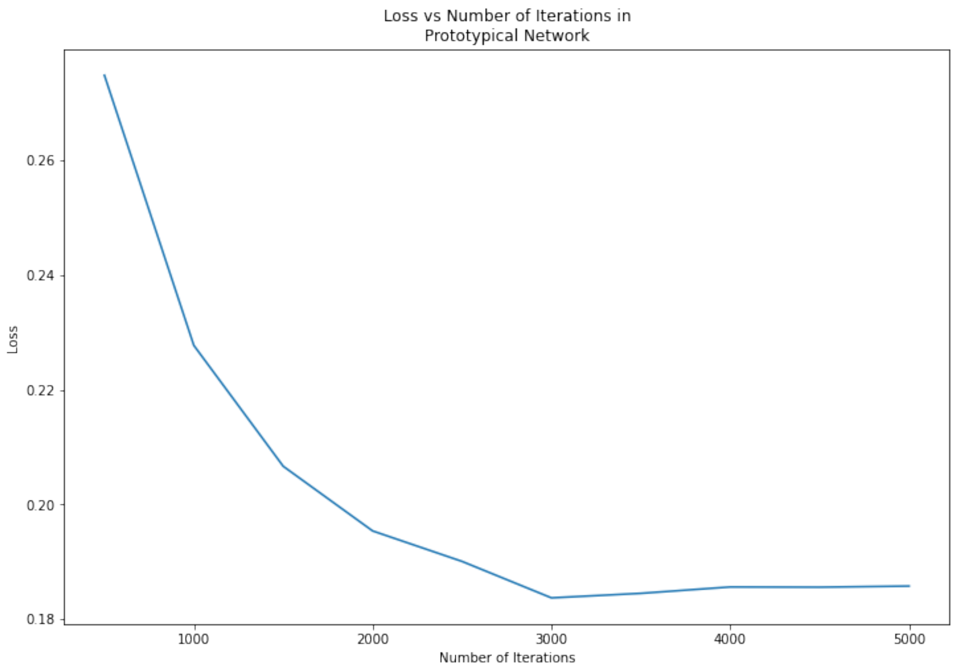


(b) Loss of Siamese network

Figure 5.3: Siamese Network Evaluation.



(a) Accuracy and F1 vs No. of Samples of Prototypical Network



(b) Loss of Prototypical network

Figure 5.4: Prototypical Network Evaluation.

Table 5.4: Evaluation of Siamese Network With Different Kernel and Filter Sizes

| **Dataset** | **Kernel** | **Filter Size** | **Accuracy** | **F1 Score** |
|---|---|---|---|---|
| CSE-CIC-IDS2017 | 2 x 2 | 32 | 90.588 | 93.986 |
| | | 64 | 91.488 | 94.331 |
| | | 128 | 90.425 | 93.415 |
| | 3 x 3 | 32 | 91.719 | 94.473 |
| | | 64 | 93.222 | 95.506 |
| | | 128 | 93.952 | 95.925 |
| | **4 x 4** | 32 | 90.374 | 93.401 |
| | | **64** | **94.131** | **96.113** |
| | | 128 | 93.665 | 95.724 |
| CSE-CIC-IDS2018 | 2 x 2 | 32 | 94.241 | 93.027 |
| | | 64 | 94.306 | 93.912 |
| | | 128 | 94.336 | 93.899 |
| | 3 x 3 | 32 | 94.103 | 93.209 |
| | | 64 | 93.431 | 92.995 |
| | | 128 | 93.560 | 93.149 |
| | **4 x 4** | 32 | 93.774 | 93.248 |
| | | **64** | **94.357** | **93.929** |
| | | 128 | 93.651 | 93.104 |

Table 5.5: Multi-class Classification Results

| Dataset | Label | Precision | Recall | F1 Score |
|---|---|---|---|---|
| IDS2017 | Bot | 0.9191 | 0.9660 | 0.9420 |
| | DDoS | 0.9121 | 0.9409 | 0.9263 |
| | DoS GoldenEye | 0.9326 | 0.9404 | 0.9365 |
| | DoS Hulk | 0.9675 | 0.9399 | 0.9535 |
| | DoS Slowhttptest | 0.9679 | 0.9515 | 0.9597 |
| | DoS Slowloris | 0.8404 | 0.8239 | 0.8321 |
| | FTP Bruteforce | 0.8975 | 0.9933 | 0.9430 |
| | Heartbleed | 1.0000 | 1.0000 | 1.0000 |
| | Infiltration | 0.0216 | 0.8333 | 0.0422 |
| | PortScan | 0.9990 | 0.9908 | 0.9949 |
| | SSH Bruteforce | 0.9090 | 0.8360 | 0.8709 |
| | Bruteforce Web | 0.4805 | 0.3274 | 0.3895 |
| | SQL Injection | 0.0127 | 1.0000 | 0.0251 |
| | Bruteforce XSS | 0.2338 | 0.5510 | 0.3283 |
| IDS2018 | Bot | 0.9913 | 0.9896 | 0.9905 |
| | Brute Web | 0.1190 | 0.6593 | 0.2017 |
| | Brute XSS | 0.0414 | 0.9706 | 0.0794 |
| | DDoS HOIC | 0.9990 | 1.0000 | 0.9995 |
| | DDoS LOIC UDP | 0.9449 | 0.9885 | 0.9662 |
| | DDoS LOIC HTTP | 0.9993 | 1.0000 | 0.9997 |
| | DoS GoldenEye | 0.9810 | 0.9640 | 0.9725 |
| | DoS Hulk | 0.9889 | 0.9971 | 0.9930 |
| | DoS SlowHTTPTest | 0.6024 | 0.8271 | 0.6971 |
| | DoS Slowloris | 0.8415 | 0.9982 | 0.9132 |
| | Brute FTP | 0.7523 | 0.4914 | 0.5945 |
| | Infiltration | 0.9882 | 0.8892 | 0.9361 |
| | SQL Injection | 0.0127 | 0.5385 | 0.0249 |
| | Brute SSH | 0.9946 | 1.0000 | 0.9973 |

the model better fit the data. The table 5.4 represents the accuracy and F-1 score for the different combinations of kernel size and filter size of the Siamese network. From the table 5.4, it is clearly visible that our Siamese network achieved the best outcomes when the kernel size is 4 x 4 and the filter size is 64 for both datasets.
The accuracy and F-1 score for the various kernel size and filter size combinations of the prototypical network are shown in the table 5.6. The table 5.6 makes it quite evident that for both datasets, our prototypical network performed best when the kernel size was 4 x 4 and the filter size was 64.

Table 5.6: Evaluation of Prototypical Network With Different Kernel and Filter Sizes

| **Dataset** | **Kernel** | **Filter Size** | **Accuracy** | **F1 Score** |
|---|---|---|---|---|
| CSE-CIC-IDS2017 | 2 x 2 | 32 | 86.795 | 88.561 |
| | | 64 | 91.605 | 92.337 |
| | | 128 | 92.664 | 93.383 |
| | 3 x 3 | 32 | 95.019 | 95.415 |
| | | 64 | 95.486 | 95.996 |
| | | 128 | 95.169 | 95.801 |
| | **4 x 4** | 32 | 94.974 | 95.521 |
| | | **64** | **95.683** | **96.105** |
| | | 128 | 95.509 | 95.833 |
| CSE-CIC-IDS2018 | 2 x 2 | 32 | 87.631 | 88.116 |
| | | 64 | 88.405 | 88.855 |
| | | 128 | 88.455 | 88.975 |
| | 3 x 3 | 32 | 89.640 | 89.975 |
| | | 64 | 90.257 | 90.464 |
| | | 128 | 90.162 | 90.453 |
| | **4 x 4** | 32 | 90.033 | 90.303 |
| | | **64** | **90.640** | **91.001** |
| | | 128 | 90.495 | 90.751 |

By analysing the above mentioned information, we choose the combination of 4 x 4 kernel size and 64 filter size for our architecture as we get the individual best outcomes with this combination.
As we mentioned above, the ROC curve is referred to as an important metric to determine the performance of the classifiers. Thus we are using the ROC curve to see the performance of our proposed architecture. Figure 5.5 shows how our Siamese network performs during malicious data identification. From the figure, it is seen that the performance of the Siamese network doesn't vary significantly between the datasets. Additionally, the AUC score referenced in table 5.7 is very close to 1.0 signifying that the model is performing well. However, it works better for the CSE-CIC-IDS2017 dataset as the curve is more to the top left corner of the graph. This

is also shown in the AUC score.

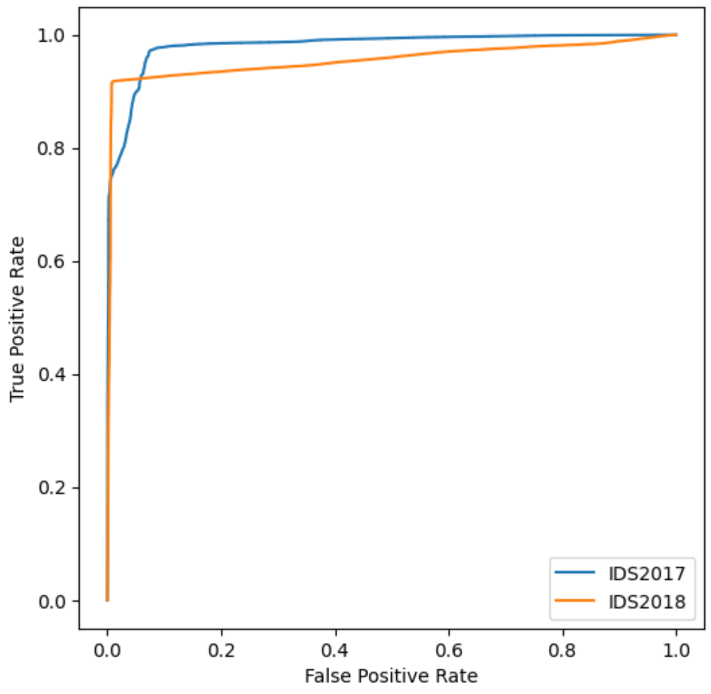
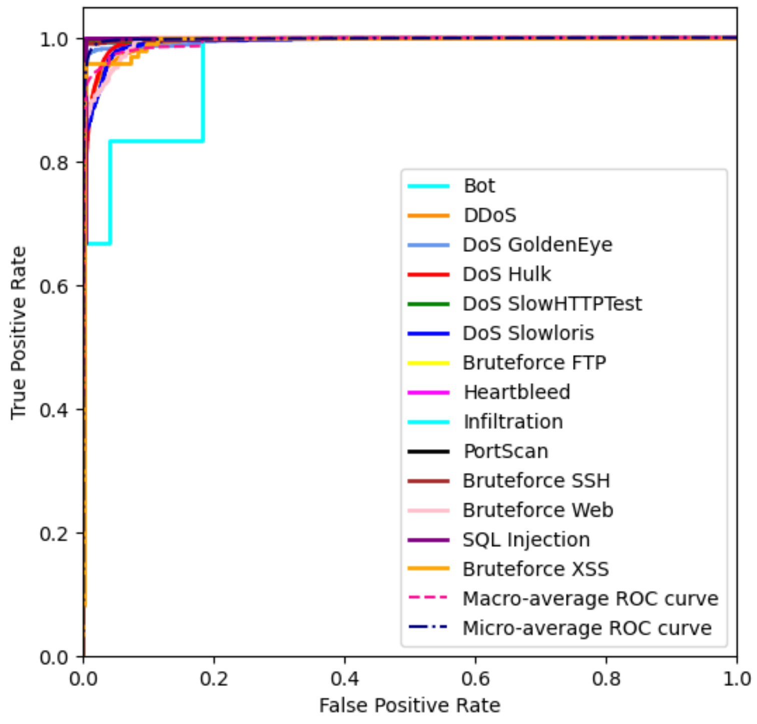


Figure 5.5: ROC curves of Siamese Network for CSE-CIC-IDS2017 and CSE-CIC-IDS2018

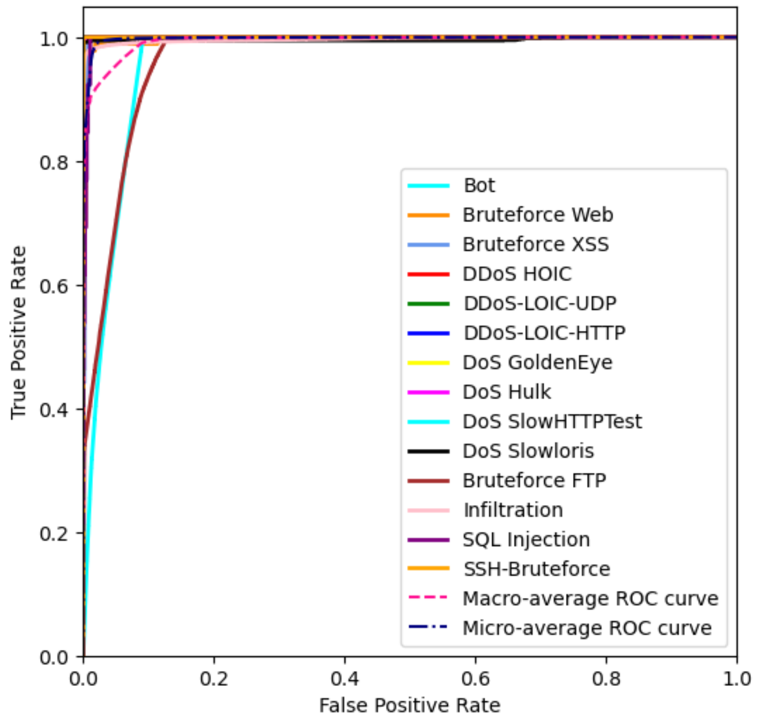
Table 5.7: AUC Scores of Siamese Network of Different Datasets

| **Dataset** | **AUC** |
|---|---|
| CSE-CIC-IDS2017 | 98.119 |
| CSE-CIC-IDS2018 | 95.575 |

ROC curves are mostly used to evaluate the effectiveness of binary classifiers. However, it takes into account two distinct approaches when it comes to multiclass classification. One vs Rest and One vs One are the two different techniques. We are considering One vs Rest to measure the performance of our Prototypical network. Here, at a time, one label is considered as positive and the rest are considered as negative. Using this, TPR and FPR are calculated and the ROC curve is plotted. To measure the overall performance of the Prototypical network, we also provide the micro and macro average of all the ROC curves.
ROC curves for CSE-CIC-IDS17 and CSE-CIC-IDS18 are displayed in figure 5.6a and figure 5.6b respectively. In figure 5.6a, it is observed that the performance of the label Infiltration is worse compared to the rest of them as evident from the figure. It is also visible in the AUC score in table 5.8.
In figure 5.6b, it is seen that the performance of SlowHTTPTest and Bruteforce FTP are worse compared to others as both the curves of these two labels get less closer to the top of the graph. Moreover, it is also observable in the AUC score in table 5.8 which is calculated from the ROC curve.



(a) One vs. Rest ROC Curves of Prototypical Network using CSE-CIC-IDS2017



(b) One vs. Rest ROC Curves of Prototypical Network using CSE-CIC-IDS2018

Figure 5.6: ROC curves of Prototypical Network for CSE-CIC-IDS2017 and CSE-CIC-IDS2018

Table 5.8: OvR AUC Scores of Prototypical Network of Different Datasets

| **Dataset** | **Label** | **OvR AUC** |
|---|---|---|
| CSE-CIC-IDS2017 | Bot | 0.999 |
| | DDoS | 0.993 |
| | DoS GoldenEye | 0.997 |
| | DoS Hulk | 0.994 |
| | DoS SlowHTTPTest | 0.999 |
| | DoS Slowloris | 0.993 |
| | Bruteforce FTP | 1.000 |
| | Heartbleed | 1.000 |
| | Infiltration | 0.963 |
| | PortScan | 0.999 |
| | Bruteforce SSH | 0.998 |
| | Bruteforce Web | 0.994 |
| | SQL Injection | 0.998 |
| | Bruteforce XSS | 0.995 |
| | Macro Average | 0.995 |
| | Micro Average | 0.998 |
| CSE-CIC-IDS2018 | Bot | 1.000 |
| | Bruteforce Web | 0.997 |
| | Bruteforce XSS | 0.998 |
| | DDoS HOIC | 1.000 |
| | DDoS-LOIC-UDP | 1.000 |
| | DDoS-LOIC-HTTP | 1.000 |
| | DoS GoldenEye | 1.000 |
| | DoS Hulk | 0.999 |
| | DoS SlowHTTPTest | 0.965 |
| | DoS Slowloris | 0.996 |
| | Bruteforce FTP | 0.967 |
| | Infiltration | 0.997 |
| | SQL Injection | 0.998 |
| | SSH-Bruteforce | 1.000 |
| | Macro Average | 0.994 |
| | Micro Average | 0.998 |

## 5.4 Comparative Study

Following details will demonstrate an overview between our work and some of the existing work in this domain. Table 5.9 also shows this comparison.

Table 5.9: Multi-class Classification Comparison with Related Work

| Related Work | Dataset | Data Used | No. of Classes | Accuracy |
|---|---|---|---|---|
| Our Hybrid Meta Approach | IDS2017 | 3000 | 14 | 95.68 |
| | IDS2018 | 3000 | 14 | 90.64 |
| Our RL Approaches | IDS2018 | 1575000 | 15 | 70 |
| CNN Based[13] | IDS2018 | 10407862 | 6 | 91.5 |
| Siamese One Shot[23] | IDS2017 | 274729 | 5 | 80.81 - 82.5 |
| Siamese Capsule[20] | IDS2017 | 3025 | 8 | 95.25 |

In our previous work, we implemented RL with two alternative strategies. Both of the models were trained with a huge amount of data to achieve approximately 70% accuracy which is not good enough as the existing models were performing better than that. Moreover, we used SMOTE to oversample the data to restrict our model from being biased towards labels with higher data samples. As a result, the dataset grows significantly. It consumed a lot of storage space and increased the processing time. Even compromising with the storage and time, we couldn't get better outcomes. Moreover, oversampled data do not always reflect real world data. The primary goal of a reinforcement learning agent is to maximize the reward, and the method uses the agent's reward calculation to turn it into a dynamic programming approach. Agents determine the total current reward by using information from the present state along with a prediction of future expected reward based on the current trajectory.The equation is as follows:

$$G(t) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \dots$$

This makes RL a dynamic programming approach where the algorithm is taking future steps into consideration before making any action. However, in classification setting, one state is completely independent of the other regardless of the action being taken as the next network traffic can be of any class. As a result we need to keep $\gamma$ close to zero, making the approach a greedy algorithm. This is another reason why our RL approach could not perform well. These constraints drove us to conduct additional research in order to develop a novel concept that could be effectively trained on modest amounts of data. Due to the fact that meta learning techniques may be utilized to train models efficiently with limited data, we therefore incorporated a hybrid meta deep learning strategy. Here, we trained our models using 3000 data samples, applying the settings mentioned in tables 5.4 and 5.6. In contrast to our earlier study, where we needed 1575000, this amount of data training data samples is incredibly minimal. We were able to classify 15 different labels in the CSE-CIC-IDS2018 with about 94% and 90% accuracy using Siamese and prototypical networks, respectively, despite using 3000 data samples.
In [13], an experiment on binary and multiclass classification for DoS attacks was also conducted by the authors. But for this task, they employed a CNN. They trained

and tested their model using about 10407862 samples of data from the CSE-CIC-IDS2018, and their binary classification accuracy was 91.5%. For 6 distinct labeled DoS attacks, they also achieved 91.5% for multiclass classification. For comparison purposes, We used their method for 15 unique labels, including Benign from the CSE-CIC-IDS 2018 dataset. For multiclass identification, their approach achieved an accuracy of about 91.69% and 90.70% f1 score, 92.34% precision and 91.69% recall.

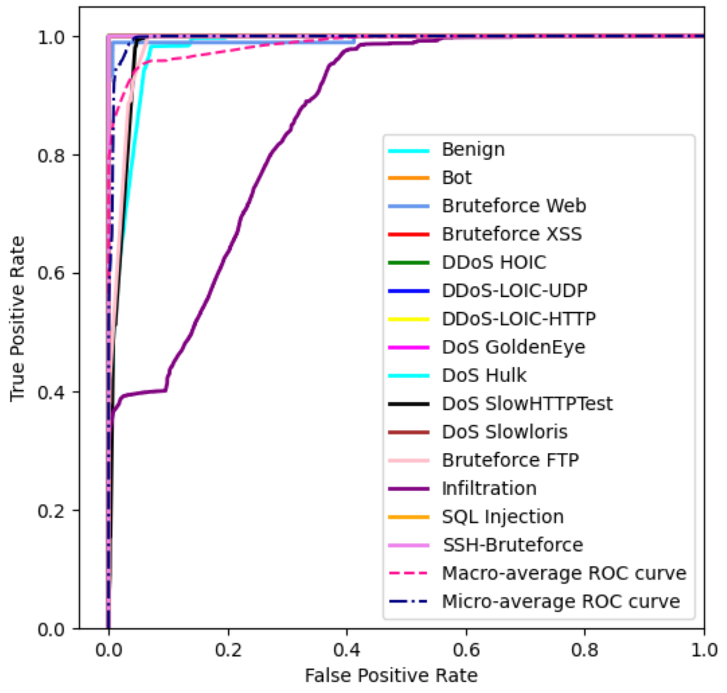


Figure 5.7: ROC curves of CNN Based Approach [13] with 15 Labels

On the other hand, with only 3000 data, we were able to reach an accuracy of 90.29%. Additionally, the f1 score, precision, and recall closely match their results. They used 10407862 samples to arrive at a conclusion where our model achieved it with 3000 samples. Our approach even outperformed them in some cases. For example, classifying infiltration using our hybrid meta learning approach, we got 98.82% precision, 88.92% recall and 93.61% f1 score whereas using CNN we got 97.72% precision, 40.30% recall and 57.07% f1 score.
We also used ROC curve and AUC score to compare the outcomes of this study [13] with our work. As we mentioned above the ROC curve provided the best evaluation picture for classification problems. Figure 5.7 shows their OvR ROC curves for 15 different labels. It is clear from the figure that Infiltration performance is worse compared to others. From table 5.10 it is clearly visible that the AUC score of infiltration is around 85% using CSE-CIC-IDS2018. However, from the table 5.8, it is evident that our proposed framework works better as in our model, AUC score for infiltration has the lowest value of only 96.3%. Moreover, the other labels that perform worse are still above 95% score. From this analysis, we can say that our work performs better in terms of AUC score which is also evident from the micro

Table 5.10: OvR AUC Scores of CNN Based Approach [13] with 15 Labels

| **Label** | **OvR AUC** |
|---|---|
| Benign | 0.979 |
| Bot | 0.999 |
| Bruteforce Web | 0.995 |
| Bruteforce XSS | 0.999 |
| DDoS HOIC | 1.0 |
| DDoS-LOIC-UDP | 1.0 |
| DDoS-LOIC-HTTP | 1.0 |
| DoS GoldenEye | 0.999 |
| DoS Hulk | 0.999 |
| DoS SlowHTTPTest | 0.984 |
| DoS Slowloris | 0.999 |
| Bruteforce FTP | 0.986 |
| Infiltration | 0.855 |
| SQL Injection | 0.999 |
| SSH-Bruteforce | 0.999 |
| Macro Average | 0.987 |
| Micro Average | 0.996 |

and macro average scores.
In [23] authors recommended a Siamese Network using one shot learning mechanism to classify cyber attacks. In total they used 274729 samples for their experiment. Among them 248607 samples are normal data and the rest are divided into DoS(Hulk), DoS(Slowloris), FTP Brute Force, SSH Brute Force. For 5 different cyber attacks, they achieved 80.81% to 82.5% accuracy using the CSE-CIC-IDS 2017 dataset. On the other hand, we took all the attacks into our consideration and utilizing the same dataset, our suggested hybrid meta learning model had an accuracy of about 94.22% for 14 distinct network labels in multiclass classification with 3000 data samples.
Authors proposed a Siamese Network based model in [20] to enhance the network intrusion detection using unbalanced training data. They presented their experiment with CSE-CIC-IDS 2017. They achieved an accuracy of about 95.25% for the 8 different multiclass classifications. However, our suggested system achieved accuracy that is very close to their results while taking into account all forms of malicious data.

# Chapter 6

## 6.1 Conclusion

In this paper, we introduced an RL method and a hybrid meta learning approach to detect malicious packet data and identify them using multiclass classification. Reviewing papers gave us the insight that there is a prominent requirement of research in secure communication because of not having enough data due to their sensitive nature. Moreover, malicious data is also very uncommon causing publicly available datasets to be highly unbalanced. To address these challenges, our primary step was introducing a model using RL technique. Here we used two different approaches called A2C and PPO. Using RL techniques we had to use a lot of data and for some classes with lower data samples we had to follow oversampling techniques. This approach also took a lot of resources and achieved an accuracy of around 70% for both the algorithms. Unfortunately, these outcomes couldn't outperform some of the state-of-the-art techniques. Thus we propose a new technique called hybrid meta deep learning. The advancement of meta learning technologies can be an excellent solution in providing a trustworthy mechanism for this. Therefore, our work attempted to address the existing security issues. This approach not only ensures a secure transmission by detecting malicious packet data but also classifies them into multiple classes. Users could use this information to have more fine-grained control over which preventative measures to take for specific attack types. Our proposed architecture doesn't need a lot of data to be trained. Hybrid meta learning techniques can train the model efficiently with a very small amount of data compared to the other existing approaches. Thus it resolves the issue of data insufficiency. Furthermore, at the time of our research, we could not find any work addressing more than 8 classes where our model was able to classify 15 distinct classes with greater than 90% accuracy. But even though the model can be trained with small amounts of data, for the data variation and to improve the model we still need data. Additionally, it is impractical to provide security against all kinds of attacks. So to address this problem, we consider introducing a federated learning mechanism as our future task which will ensure continuation of data and improve the performance of our model to detect more malicious attacks. Furthermore, incorporating attention mechanisms with the proposed framework might give a direction to the future researchers in this domain. Most importantly, we anticipated that the findings of our strategy might be an essential component of a platform for end-to-end trust networking.In conclusion, we sincerely believe that our efforts will be beneficial to solve the security crisis for the upcoming days.

## 6.2 Limitations

Like any other research study, ours has some restrictions. As we mentioned before, malicious network data is scarce. As a result, datasets that are made publicly available are highly unbalanced.Thus, we had limitations in training models with diverse data samples. Additionally, people show unwillingness in sharing their private data because of confidentiality issues. As a result, we were unable to assess the performance of our model by training it on more recent real-world traffic data. Furthermore, due to time restrictions, we could not evaluate the performance of our model using raw network packets.The fact that meta learning is a relatively new field of study is another problem we had to face. As a consequence of this, learning about this topic required a lot of effort on our part.

## 6.3 Future Work

For the time constraints of our research, we weren't able to incorporate more techniques to see how much they deal with our recent work. So we consider them as our future thoughts. The following section might help the researcher to get a future direction in this domain. Federated Learning can be added to maintain the continuation of data. It might bring the variation of data and improve the performance of the model. Additionally, attention mechanisms can also be incorporated. It will allow the model to focus on more important features further boosting performance. Moreover, an end to end trust networking platform can be formed. The outcomes from the hybrid meta learning approach can be used to construct a trusted platform.

# Bibliography

[1] C. E. Metz, “Basic principles of roc analysis,” *Seminars in Nuclear Medicine*, vol. 8, no. 4, pp. 283–298, 1978, ISSN: 0001-2998. DOI: https://doi.org/10.1016/S0001-2998(78)80014-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001299878800142.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002. DOI: 10.1613/jair.953. [Online]. Available: https://doi.org/10.1613%5C%2Fjair.953.

[3] R. Bie, X. Jin, C. Chen, C. Xu, and R. Huang, “Meta learning intrusion detection in real time network,” vol. 4668, Sep. 2007, pp. 809–816, ISBN: 978-3-540-74689-8. DOI: 10.1007/978-3-540-74690-4_82.

[4] V. Mnih, A. P. Badia, M. Mirza, *et al.*, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, Jun. 2016, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017. DOI: 10.48550/ARXIV.1707.06347. [Online]. Available: https://arxiv.org/abs/1707.06347.

[6] S. Ravichandiran, *Hands-On Meta Learning with Python*. Birmingham, England: Packt Publishing, Dec. 2018.

[7] R. Yamashita, M. Nishio, R. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” *Insights into Imaging*, vol. 9, Jun. 2018. DOI: 10.1007/s13244-018-0639-9.

[8] J. Kim, Y. Shin, and E. Choi, “An intrusion detection model based on a convolutional neural network,” *Journal of Multimedia Information System*, vol. 6, pp. 165–172, Dec. 2019. DOI: 10.33851/JMIS.2019.6.4.165.

[9] S. Niknam, H. S. Dhillon, and J. H. Reed, *Federated learning for wireless communications: Motivation, opportunities and challenges*, 2019. DOI: 10.48550/ARXIV.1908.06847. [Online]. Available: https://arxiv.org/abs/1908.06847.

[10] D. Sun, Z. Wu, Y. Wang, Q. Lv, and B. Hu, “Risk prediction for imbalanced data in cyber security : A siamese network-based deep learning classification framework,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8852030.

[11] J. Yang, S. He, Y. Xu, L. Chen, and J. Ren, "A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks," *Sensors*, vol. 19, no. 4, 2019, ISSN: 1424-8220. DOI: 10.3390/s19040970. [Online]. Available: https://www.mdpi.com/1424-8220/19/4/970.

[12] R. Kantola, "Trust networking for beyond 5g and 6g," Mar. 2020. DOI: 10.13140/RG.2.2.31564.16000.

[13] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "Cnn-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, p. 916, Jun. 2020. DOI: 10.3390/electronics9060916.

[14] S. Kim, "Incentive design and differential privacy based federated learning: A mechanism design perspective," *IEEE Access*, vol. 8, pp. 187 317–187 325, 2020. DOI: 10.1109/ACCESS.2020.3030888.

[15] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, "Federated learning for 6g communications: Challenges, methods, and future directions," *China Communications*, vol. 17, no. 9, pp. 105–118, 2020. DOI: 10.23919/JCC.2020.09.009.

[16] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222 310–222 354, 2020. DOI: 10.1109/ACCESS.2020.3041951.

[17] J. Wang and Y. Zhai, "Prototypical siamese networks for few-shot learning," in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2020, pp. 178–181. DOI: 10.1109/ICEIEC49280.2020.9152261.

[18] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate dos attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43 920–43 943, 2020. DOI: 10.1109/ACCESS.2020.2976609.

[19] D. Park, S. Kim, H. Kwon, D. Shin, and D. Shin, "Host-based intrusion detection model using siamese network," *IEEE Access*, vol. 9, pp. 76 614–76 623, 2021. DOI: 10.1109/ACCESS.2021.3082160.

[20] Z.-M. Wang, J.-Y. Tian, J. Qin, H. Fang, and L.-M. Chen, "A few-shot learning-based siamese capsule network for intrusion detection with imbalanced training data," *Computational Intelligence and Neuroscience*, vol. 2021, pp. 1–17, Sep. 2021. DOI: 10.1155/2021/7126913.

[21] M. Wazid, A. K. Das, S. Shetty, P. Gope, and J. J. P. C. Rodrigues, "Security in 5g-enabled internet of things communication: Issues, challenges, and future research roadmap," *IEEE Access*, vol. 9, pp. 4466–4489, 2021. DOI: 10.1109/ACCESS.2020.3047895.

[22] G. Cheng, L. Cai, C. Lang, *et al.*, "Spnet: Siamese-prototype network for few-shot remote sensing image scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2022. DOI: 10.1109/TGRS.2021.3099033.

[23] H. Hindy, C. Tachtatzis, R. Atkinson, *et al.*, "Leveraging siamese networks for one-shot intrusion detection model," *Journal of Intelligent Information Systems*, pp. 1–30, Nov. 2022. DOI: 10.1007/s10844-022-00747-z.

[24] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, “Machine and deep learning solutions for intrusion detection and prevention in iots: A survey,” *IEEE Access*, vol. 10, pp. 121 173–121 192, 2022. DOI: 10.1109/ACCESS.2022.3220622.

[25] S. Y. Khamaiseh, D. Bagagem, A. Al-Alaj, M. Mancino, and H. W. Alomari, “Adversarial deep learning: A survey on adversarial attacks and defense mechanisms on image classification,” *IEEE Access*, vol. 10, pp. 102 266–102 291, 2022. DOI: 10.1109/ACCESS.2022.3208131.

[26] S. Mo, Z. Sun, and C. Li, “Siamese prototypical contrastive learning,” *CoRR*, vol. abs/2208.08819, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.08819.

[27] S. Neupane, J. Ables, W. Anderson, *et al.*, “Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities,” *IEEE Access*, vol. 10, pp. 112 392–112 415, 2022. DOI: 10.1109/ACCESS.2022.3216617.