**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL**

# ENHANCING BOTNET DETECTION USING FEDERATED LEARNING IN IOT NETWORKS

**M.Sc. THESIS**

**Nilüfer USLAN**

**Department of Computer Engineering**

**Computer Engineering Programme**

**JUNE 2025**

**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL**

# ENHANCING BOTNET DETECTION USING FEDERATED LEARNING IN IOT NETWORKS

**M.Sc. THESIS**

**Nilüfer USLAN**
**(504221526)**

**Department of Computer Engineering**

**Computer Engineering Programme**

**Thesis Advisor : Prof. Dr. Şerif BAHTİYAR**

**JUNE 2025**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# IOT AĞLARINDA FEDERE ÖĞRENME YÖNTEMİNİ KULLANARAK BOTNET TESPİTİNİN GELİŞTİRİLMESİ

**YÜKSEK LİSANS TEZİ**

**Nilüfer USLAN**
**(504221526)**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Bilgisayar Mühendisliği Programı**

**Tez Danışmanı : Prof. Dr. Şerif BAHTİYAR**

**HAZİRAN 2025**

Nilüfer USLAN, a M.Sc. student of ITU Graduate School student ID 504221526, successfully defended the thesis entitled "ENHANCING BOTNET DETECTION USING FEDERATED LEARNING IN IOT NETWORKS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

| | | |
|---|---|---|
| **Thesis Advisor :** | **Prof. Dr. Şerif BAHTİYAR**<br>Istanbul Technical University | ..... |
| **Jury Members :** | **Dr. Öğr. Üyesi Mehmet Tahir SANDIKKAYA**<br>Istanbul Technical University | ..... |
| | **Prof. Dr. Alptekin KÜPÇÜ**<br>Koç University | ..... |

**Date of Submission : 28 May 2025**
**Date of Defense : 25 June 2025**

***To my family,***

**FOREWORD**

I would like to express my sincere gratitude to my advisor, Prof. Şerif Bahtiyar, for his undying trust in me. I would also like to express my sincere appreciations to Ömür Fatmanur Erzurumluoğlu and Burcu Sönmez for their generous support.

June 2025

Nilüfer USLAN
Computer Engineer

**TABLE OF CONTENTS**

**Page**

**FOREWORD** ..... **ix**
**TABLE OF CONTENTS** ..... **xi**
**ABBREVIATIONS** ..... **xiii**
**SYMBOLS** ..... **xv**
**LIST OF TABLES** ..... **xvii**
**LIST OF FIGURES** ..... **xix**
**SUMMARY** ..... **xxi**
**ÖZET** ..... **xxiii**
**1. INTRODUCTION** ..... **1**
**2. BACKGROUND** ..... **3**
2.1 Problem Statement ..... 3
2.2 Botnets ..... 4
2.2.1 Architecture ..... 4
2.2.1.1 Centralized ..... 5
2.2.1.2 Decentralized ..... 5
2.3 IoT Security ..... 5
2.4 Federated Learning ..... 6
**3. LITERATURE REVIEW** ..... **7**
**4. FEDERATED LEARNING FRAMEWORK FOR IDS SYSTEMS** ..... **11**
4.1 N-BaIoT Dataset ..... 11
4.1.1 Preprocessing ..... 12
4.2 Feature Selection ..... 13
4.2.1 Performance on neural networks ..... 13
4.2.1.1 Classification by label ..... 14
4.2.1.2 Classification by botnet type ..... 15
4.2.1.3 Classification by attack code ..... 15
4.3 Synthetic Data Generation ..... 16
4.3.1 CTGANs ..... 16
4.3.2 Evaluation of synthetic data ..... 16
4.3.2.1 Feature distribution comparison ..... 17
4.3.2.2 Detection performance on DNN ..... 17
4.3.3 Metric Comparison ..... 17
4.4 Neural Networks ..... 17
4.4.1 Hyperparameter tuning ..... 17
4.4.2 Evaluation of DNN model ..... 18
4.5 Federated Learning ..... 18
4.5.1 Proposed architecture ..... 18
4.5.1.1 Model Initialization ..... 20
4.5.1.2 Algorithm ..... 20

4.5.2 Experiments ..... 20
4.5.2.1 Parameter optimization for federated learning ..... 20
4.5.2.2 Synthetic data generation integration point ..... 21
4.5.3 Results ..... 24
**5. CONCLUSION ..... 27**
**REFERENCES ..... 31**
**APPENDIX ..... 35**
APPENDIX A ..... 37
APPENDIX B ..... 41
APPENDIX C ..... 43
**CURRICULUM VITAE ..... 47**

**ABBREVIATIONS**

| | |
|---|---|
| **IoT** | **:** Internet of Things |
| **C2** | **:** Command And Control |
| **DDoS** | **:** Distributed Denial of Service |
| **BP** | **:** Backpropagation |
| **DNN** | **:** Deep Neural Network |
| **MLP** | **:** Multi Layer Perceptron |
| **CTGAN** | **:** Conditional Tabular Generative Adversarial Network |
| **R** | **:** Number of Training Rounds |
| **C** | **:** Number of Random Client in Each Training Rounds |
| **IID** | **:** Identical and Independent Distribution |
| **non-IID** | **:** Non Identical and Independent Distribution |
| **CTGAN** | **:** Conditional Tabular Generative Adversarial Network |

**SYMBOLS**

| | |
|---|---|
| $\mu$ | **:** Mean of the feature in the training data |
| $\sigma$ | **:** Standard deviation |
| **x** | **:** Sample data |
| **tw** | **:** Time window |

**LIST OF TABLES**

**Page**


**Table 3.1** : Studies using N-BaIoT dataset..... **8**
**Table 4.1** : Overview of IoT devices in N-BaIoT dataset. ..... **11**
**Table 4.2** : Sample count by botnet type after dropping duplicates. ..... **12**
**Table 4.3** : Performance comparison of DNN model with different feature sets.. **12**
**Table 4.4** : Performance comparison of DNN model with different feature sets.. **13**
**Table 4.5** : Selected feature names. ..... **13**
**Table 4.6** : Performance comparison of DNN model with different feature sets.. **14**
**Table 4.7** : Performance comparison of DNN model with and without the synthetic data. ..... **17**
**Table 4.8** : Hyperparameter tuning for DNN. ..... **17**
**Table 4.9** : Best parameter for training round (R), client per round (C), and epoch (10). ..... **21**
**Table A.1** : KS Statistics results of each feature from synthetic data by device id. **38**
**Table A.2** : Wassertein Distance results of each feature from synthetic data by device id. ..... **39**
**Table A.3** : KL Divergence results of each feature from synthetic data by device id. ..... **40**

**LIST OF FIGURES**

**Page**


**Figure 2.1** : Centralized botnet architecture. ..... **5**
**Figure 2.2** : Decentralized botnet architecture. ..... **6**
**Figure 4.1** : Confusion matrix (115 features). ..... **14**
**Figure 4.2** : Confusion matrix (10 features). ..... **14**
**Figure 4.3** : Performance metrics - Botnet type (115 features). ..... **15**
**Figure 4.4** : Performance metrics - Botnet type (10 features). ..... **15**
**Figure 4.5** : Confusion matrix - Botnet type (10 features). ..... **15**
**Figure 4.6** : Confusion matrix - Botnet type (115 features). ..... **16**
**Figure 4.7** : DNN model performance on a balanced dataset over epochs. ..... **18**
**Figure 4.8** : Device-based accuracy of tuned DNN model. ..... **18**
**Figure 4.9** : Proposed architecture. ..... **19**
**Figure 4.10** : Preliminary step for OutFed architecture. ..... **22**
**Figure 4.11** : NoSyn architecture ..... **23**
**Figure 4.12** : NoSyn experiment runtime for 10 rotations. ..... **24**
**Figure 4.13** : InFed experiment runtime for 10 rotations. ..... **24**
**Figure 4.14** : OutFed experiment runtime for 10 rotations. ..... **25**
**Figure 4.15** : Average accuracy for each experiment NoSyn, InFed, and OutFed. .. **25**
**Figure B.1** : Confusion matrix - Attack code type (115 features). ..... **41**
**Figure B.2** : Confusion matrix - Attack code type (10 features). ..... **41**
**Figure C.1** : Distribution comparison of HH_jit_L0.1_weight. ..... **43**
**Figure C.2** : Distribution comparison of HH_L1_std. ..... **43**
**Figure C.3** : Distribution comparison of HH_jit_L0.01_mean. ..... **43**
**Figure C.4** : Distribution comparison of MI$_d$*ir_L*0.01*_mean*. ..... **44**
**Figure C.5** : Distribution comparison of H_L0.01_weight. ..... **44**
**Figure C.6** : Distribution comparison of H_L1_variance. ..... **44**
**Figure C.7** : Distribution comparison of H_L3_weight. ..... **44**
**Figure C.8** : Distribution comparison of HpHp_L0.01_weight. ..... **45**
**Figure C.9** : Distribution comparison of HpHp_L3_std. ..... **45**
**Figure C.10** : Distribution comparison of HpHp_L5_weight. ..... **45**

## ENHANCING BOTNET DETECTION USING FEDERATED LEARNING IN IOT NETWORKS

## SUMMARY

A botnet can be defined as a network of compromised devices, usually controlled by a malicious actor. Botnets are utilized to launch cyberattacks, including Distributed Denial of Service (DDoS) attacks, theft of sensitive financial data, and cryptocurrency mining. These compromised devices listen for commands from the malicious actor and execute them when they are received, with the mechanism often provided by command-and-control (C2) servers. The true strength of a botnet lies in its scale, as it can consist of millions of compromised devices working in unison. Therefore, the rapid growth in the usage of Internet of Things (IoT) devices has significantly increased the presence of botnet threats, as many of them lack strong security. Devices such as smart home devices, wearable activity trackers, security cameras, and routers are particularly vulnerable due to their widespread internet connectivity and minimal user oversight. The reliance of botnets on extensive networks of compromised devices renders them particularly well-suited to exploit the expanding IoT landscape. Consequently, the ability to detect botnets is crucial for preventing cyberattacks and protecting sensitive data. However, traditional security methods often fail in IoT environments, making advanced detection techniques necessary. Fast and accurate detection is essential to minimize damage. Fast detection will allow us to prevent botnet attacks before any damage is done. This study explores techniques such as machine learning and optimized feature selection, enhancing botnet detection, boosting performance for rapid detection, improving accuracy, and strengthening IoT security against botnet threats. Our dataset N-BaIoT has Mirai and Gafgyt infected network traffic statistics as well as benign traffic from 9 different IoT devices. Since the dataset is imbalanced across the devices we have used CTGAN to generate synthetic data to balance the dataset. We have proposed a federated learning architecture to enhance botnet detection. The server has a global model that is aggregated with the updates coming from device local models. Using FedAvg, we have updated the global model in each training round R, where C clients were randomly chosen and updates were received from them. Every client has its own local data, local DNN model, and local GAN model. Clients train their GAN model with the local data and continuously retrain with the new incoming traffic. In each round, they used the GAN to generate synthetic data to resolve the class imbalance issue and low sample count. We have also used a hybrid feature selection method and have selected the 10 most important features. This is important since IoT devices have limited resources and applying dimensionality reduction helps to address this constraint. We have evaluated the generated synthetic data according to several metrics and measured with the DNN model to see the effect of synthetic data on accuracy. Overall, the proposed model has its own advantages, such as preserving privacy and balancing classes, which resulted in better performance with coordinated learning.

# IOT AĞLARINDA FEDERE ÖĞRENME YÖNTEMİNİ KULLANARAK BOTNET TESPİTİNİN GELİŞTİRİLMESİ

## ÖZET

Botnetler, genellikle kötü niyetli bir aktör (botmaster) tarafından kontrol edilen, güvenliği ihlal edilmiş veya ele geçirilmiş cihazlardan oluşan bir ağ olarak tanımlanabilir. Bu cihazlar kötü niyetli aktörden gelen komutları dinler ve genellikle komuta ve kontrol (C2) sunucuları tarafından sağlanan mekanizma ile aldıkları bu komutları yerine getirir. En bilinen kullanım amaçları arasında Dağıtılmış Hizmet Engelleme (DDoS) saldırıları, hassas finansal verilerin çalınması ve kripto para madenciliği gibi siber saldırılar vardır. Botnetlerin bu kadar etkili olmasının temel sebebi ölçeğidir, çünkü birlikte çalışan milyonlarca cihazdan oluşabilir. Bu da özellikle DDoS saldırılarında hedefe gönderilen ağ trafiği yükünün çok büyük boyutlara ulaşmasını sağlayabilir. Bu nedenle, son zamanlarda popülerleşen ve çok çeşitli alanlarda kullanılan Nesnelerin İnterneti (IoT) cihazlarının sayısındaki artış ve bu cihazların güvenlik zaafiyetlerinin çok olması ile birlikte botnet tehditlerinin varlığı önemli ölçüde artmıştır. Akıllı ev cihazları, robot süpürgeler, giyilebilir aktivite ve sağlık durumu takip cihazları, güvenlik kameraları ve yönlendiriciler gibi IoT cihazları doğaları gereği sürekli internete bağlı olma durumları ve bazılarının kullanıcı gözetiminde olmaması nedeni ile savunmasızdır. Botnetlerin düşük güvenlikli ve çok sayıda cihazları talep etmesi sebebiyle, artan IoT cihazları da botnetler için çok verimli bir ortam oluşturmaktadır. Sonuç olarak, IoT ağlarında botnetlerin tespit edilebilmesi, siber saldırıların önlenmesi ve hassas verilerin korunması için çok önemlidir. Bilgisayarlarda işlemci ve hafıza gibi kaynakları göz ardı edilemeyecek kadar fazla kullanan antivirüs yazılımları, IDS sistemleri veya geleneksel güvenlik yöntemlerinin IoT cihazları özelinde kullanılması mümkün olmamaktadır. Bunun sebebi çoğunlukla IoT cihazlarının kısıtlı kaynaklarının (işlem gücü ve bellek kapasitesi) olmasıdır. Bu durumda hasar almamak için hızlı ve etkili bir tespit mekanizması kurmak gerekir. Makine öğrenmesi yöntemlerinin de bu amaç için sıklıkla tercih edildiğini görüyoruz. Çünkü botnetlerin sabit ve tahmin edilebilir davranışları olmadığı için ağ trafiğindeki desenleri saptamak, hem yeni çıkan botnetlere karşı davranış tahmini yapabilmeyi sağlayacaktır hem de IoT gibi küçük cihazlardaki geleneksel yöntemlere karşı daha etkili bir yöntem olarak karşımıza çıkacaktır.

Evlerde, şirketlerde ve endüstriyel ortamlarda giderek artan IoT cihazı sayısı, botnetler gibi güvenlik tehditlerinin yönetimini her geçen gün daha da zorlaştırmaktadır. Bu cihazlar genellikle sınırlı işlem gücüne sahiptir ve çevrelerindeki ağın tamamını göremezler; bu da saldırıları kendi başlarına tespit edip engellemelerini zorlaştırır. Öte yandan, tüm verilerin analiz amacıyla merkezi bir sunucuya gönderilmesi hem gizlilik endişelerine yol açmakta hem de cihaz sayısı arttıkça bu yöntemin ölçeklenebilirliğini azaltmaktadır. Bu nedenle, cihazların hem yerel düzeyde olağandışı davranışları tespit edebilmesi hem de diğer cihazlarla iş birliği yaparak daha güçlü ve koordineli bir savunma oluşturabilmesi gerekmektedir. Bireysel cihazlar tehditleri engellemek için

yeterince güçlü olmasa da, bu görevi edge gateway, saldırı engelleme sistemleri gibi merkezi denetleyiciler üstlenebilir. Bu sistemler zararlı ağı trafiğini durdurabilir, enfekte cihazları izole edebilir ve tüm ağ genelinde güvenlik politikaları uygulayabilir. Eğer birçok cihazdan elde edilen veriler, kullanıcı gizliliğini koruyacak biçimde birleştirilebilirse, bu durum küresel tehdit algılama kapasitesini de artırarak hizmet sağlayıcıların veya altyapı operatörlerinin daha etkili yanıt vermesini sağlar. Bu sorunun çözümü, farklı türde IoT ağlarında çalışabilen, esnek ve gizliliği koruyan bir yaklaşım gerektirir. IoT cihazlarının, tehdit algılaması için yerel düzeyde kendi makine öğrenimi modellerine sahip olduğu bir algılama sistemi, bu cihazlar uç nokta (edge) cihazları olarak ele alınarak geliştirilebilir. federatif öğrenmenin bu tür sistemlerde etkili olmasının temel nedeni, özellikle dağıtık yapıya sahip botnet saldırılarının büyük ölçekte tespit edilebilmesidir. Çünkü federatif öğrenme, IoT cihazlarının birbirlerinden öğrenmesine olanak tanır. Bu sayede cihazlar, şüpheli ağ trafiğini anomali veya zararlı olarak işaretleyerek gerekli önlemlerin alınması için bu bilgiyi gateway cihazları veya bulunduğu networkün merkezi saldırı engelleme sistemlerine iletebilir.

Bir diğer mesele ise IoT cihazlarının hassas verilere sahip olabilecek olanları, evlerimizde robot süpürgeler gibi, konum veya görüntü içeren, sağlık verilerimizi monitör eden cihazlardaki gibi hassas verileri taşıyan cihazları ele alacak olursak veriyi eğitmek için bile olsa cihaz dışına çıkarmamamız gerekir. Bu esnada Federe öğrenme yöntemi devreye giriyor. Bu yöntem ile cihaz üzerindeki lokal modelin cihaz verisi ile eğitilmesi sonrasında merkezi bir yerdeki modele veriyi göndermek yerine sadece lokal modelin güncel ağırlık parametrelerini paylaşmak yetiyor. Böylece merkezi model tüm cihazlardan gelen güncel parametreleri FedAvg yöntemi ile birleştirip güncel modeli cihazlara geri gönderiyor. Bu sayede bir cihazın öğrendiği deseni diğer cihazlara da aktarmış oluyorsunuz. Federe öğrenme sadece veri mahremitenin korumakla kalmayacak aynı zamanda işbirlikçi bir yaklaşım ile cihazların daha etkili öğrenmesini sağlayacaktır.

Cihazlardaki verilerin her zaman yeterli olmayacağı durumu da var. Aynı zamanda sınıflandırmanın eğilimli olmaması için de dengeli dağılmamış verileri dengelemek için sentetik veri üreterek bu sorunun üstesinden geldik. Tablosal verilerden spesifik veri etiketine göre sentetik veri üretebilen CTGAN modelin kullanarak cihaz tabanlı dengeli dağılamamış verileri dengeleyerek hem cihaz tabanlı doğruluk oranını artırdık hem de merkezi modelin diğer cihazlarda eğilimli sonuçlar vermemesini sağladık.

Federe öğrenme aşamasında merkezi modeli taşıyan sunucu belli bir sürede tekrarlayan öğrenme süreçlerine girer. Daha önceden belirlenmiş olan sayı olan R (döngü sayısı) ve C (her döngü için seçilen client sayısı) değerleri bu süreci başlatır. Her döngüde rastgele bir client seti seçerek bu cihazlara tanımladığı modeli yollar. Seçilen cihazların kendi ürettikleri sentetik veri ve kendilerine ait lokal verileri ile modeli eğittikten sonra modelin ağırlık parametrelerini merkezi sunucu toplayarak yine daha önceden belirlenmiş bir birleştirme fonksiyonu ile bunları merkezi modelin parametreleri ile birleştirir. Biz bu çalışmada çok sıklıkla kullanılan ve iyi performans gösteren FedAvg algoritmasını kullandık. Bu algoritma cihazlardan topladığı parametrelerin ortalamalarını alır ve merkezi modeli bu parametreler ile günceller. Bu şekilde R defa tekrarlayan döngüler boyunca bu işlemler bu şekilde devam eder.

Kullandığımız N-BaIot veri seti Mirai ve Bashlite gibi iki tane büyük saldırılar ile tanınan botnet zararlısı ile enfekte edilmiş cihazlardan oluşur. Yaklaşık 7 milyon veriden tekrarlayan veriler temizlendiğinde 2,5 milyon kadar veri elde edilmiştir. Bu verilerin 500 bin kadarı temiz trafikten oluşur. 1,5 milyon kadarı Mirai ve 300 bin kadarı Bashlite yani Gafgyt diye adlandırılan botnete ait olan trafiği içermektedir. Verilerin yeniden boyutlandırılması işlemi için standart ölçeklendirme yöntemleri kullanılıp daha istikrarlı bir sınıflandırma yapılmıştır. Verilerin IoT cihazlarında eğitilmesi için büyük veriler olduğunu düşündüğümüz için bu verilerden en önemli özellikleri seçerek verilerin boyutunun azaltılması hedeflenmiştir. Bunun için öncelikle düşük varyansa sahip özelliklerin elenmesi sağlanmış olup sonrasında ilişkili olan özellik setleri belirlenmiştir. Korelasyonda olan özellikler birbiri hakkında bilgi içerdiğinden her set içinden bir özellik kullanmak yeterli olacaktır. Her setten en yüksek *gini importance* ve *mutual information* değerlerine sahip olan özellik seçilerek 115 özellikten en etkili 10'u tespit edilmiştir. DNN modelimizin bu 10 özellik ile eğittiği modelin 115 özellikle eğittiği model kadar yüksek performans sergilediği gözlemlenmiştir.

Gafgyt ve yararlı trafiğin görece daha az bulunduğu veri setiyle eğitilen DNN modelinin özellikle gafgyt tespitinde daha düşün performans gösterdiği görüldüğü için sonraki aşamalarda bu verilei dengelemek için cihazlar sentetik veri üretmiştir. Sentetik veriler üretilirken kullanılan CTGAN modelinin performansı Kolmogorov-Smirnov (KS) statistic, Wasserstein Distance ve Kullback–Leibler (KL) divergence ölçüm metrikleri kullanılarak ve aynı zamanda modelimizdeki performanslarına göre değerlendirilmiştir.

Önerdiğimiz federe öğrenme sistemi ile hem cihazlardaki ver mahremiyeti korunmuş hem de ortak bir öğrenme sağlanarak cihazların tek başına elde edebildiği doğruluk değerlerinden daha yüksek değerler elde edilmiştir. Cihazların sürekli olarak CTGAN modellerini hem yeni trafik verisi ile hem de var olan veri seti ile eğitmesi ile beraber kendi verisini dengelemesi de modellerin doğruluk değerlerini artırmıştır.

Bu çalışmada IoT ağlarında federe öğrenmenin nasıl etkili kullanılabileceğini anlattık. Özellik seçimi, sentetik veri üretimi ve önerdiğimiz sistem mimarisi ile mahremiyeti koruyarak ortak öğrenmenin nasıl sağlanacağını gösterdik.

# 1. INTRODUCTION

A botnet can be defined as a network of compromised devices, usually controlled by a malicious actor. Botnets are utilized to launch cyberattacks, including Distributed Denial of Service (DDoS) attacks, theft of sensitive financial data, and cryptocurrency mining (Suchetha and Pushpalatha, 2024). These compromised devices listen for commands from the malicious actor and execute them when they are received, with the mechanism often provided by command-and-control (C2) servers. The true strength of a botnet lies in its scale, as it can consist of millions of compromised devices working in unison. This leads to a substantial computing infrastructure. Therefore, extensive utilization of Internet of Things (IoT) devices has significantly increased the presence of botnet threats since many of them lack strong security (Hamid et al., 2021). Devices such as smart home devices, wearable activity trackers, security cameras, and routers are particularly vulnerable due to their widespread internet connectivity and minimal user oversight. The reliance of botnets on extensive networks of compromised devices renders them particularly well-suited to exploit the expanding IoT landscape. Consequently, the ability to detect botnets is crucial for preventing cyberattacks and protecting sensitive data. However, traditional security methods often fail in IoT environments, making advanced detection techniques necessary. Fast and accurate detection is essential to minimize damage. Fast detection will allow us to prevent botnet attacks before any damage is done. This study explores techniques such as machine learning and optimized feature selection, enhancing botnet detection, boosting performance for rapid detection, improving accuracy, and strengthening IoT security against botnet threats.

Despite the existence of conventional detection methods, machine learning has emerged and been used widely. There are several reasons for this. First, IoT devices have low computational capabilities, therefore, low resource usage is required by the machine learning models since there is huge amount of botnet traffic. So the feature selection has a crucial role to make the data smaller in dimensions (Rathod et al., 2024). Secondly, there are lots of IoT devices that faced a botnet malware, and

collaborative learning can be used which is another machine learning method called federated learning. But there are some challenges on federated learning where a poisoned model can lead biased classification (Saveetha et al., 2024).

GANs has become usefull for these tasks also. Since the rareness of some attacks can lead low sample counts in dataset leading the biased classification against this attack. To detect more clearly to those which has low occurance synthetic data generation is a method to balance the datasets where the unbalanced dataset impact the model in a negative way (Lim and Park, 2024). This actually trains two different model which are generator and a discriminator. In this way GANs can learn to discriminate real and synthetic data and uses this knowledge to generate more realistic data. Using GANs and Federated Learning resulted a better results in this study with the proposed architecture.

## 2. BACKGROUND

In this section, there will be some introduction to the core concepts that are used in this thesis. Since machine learning methods are used in this study, understanding botnets and their working mechanisms may help one to understand the feature selection process and requirements for the ideal performance values.

### 2.1 Problem Statement

The growing number of IoT devices in homes, companies, and industrial settings has made it increasingly difficult to manage security threats, like botnets. These devices often have limited processing power and cannot see much of the network around them, making it difficult for them to detect and respond to attacks on their own. At the same time, sending all their data to a central server for analysis raises privacy concerns and does not scale well as the number of devices increases. There is a clear need for a solution that enables devices to spot unusual behavior locally, while also working together with other devices to form a stronger, more coordinated defense. Since individual devices may not be powerful enough to block threats on their own, more capable systems, such as edge gateways or central controllers, should take over this role [6]. These systems can stop harmful traffic, isolate infected devices, and apply security rules across the network. If insights from many devices can be combined in a way that protects user privacy, this can also improve global threat detection and allow service providers or infrastructure operators to respond more effectively. Solving this challenge requires a flexible, privacy-preserving approach that works across different types of IoT networks.

A detection system where IoT devices have their own local machine learning models for detection can be modified by considering the IoT devices as IoT edge devices [6]. The main idea that federated learning will be useful for such detection systems is that its large scale, especially the distributed botnet attacks, can be detected since federated learning will allow IoT devices to learn from each other. These devices can then send flagged network traffic information to the gateway level for any actions.

## 2.2 Botnets

(Lim and Park, 2024) A botnet can be defined as a network of compromised devices, usually controlled by a malicious actor remotely. Compromised devices infected with botnet malware listen to commands given by the malicious actor. This controlling mechanism can vary in terms of client-server architectures.

A botnet, short for robot network, is a network of compromised computers or devices that a malicious actor controls remotely. These compromised devices, infected with malware, endow the attacker with a range of capabilities. Botnets can execute DDoS attacks to disrupt online services, facilitate spam and phishing campaigns on a large scale, steal sensitive data, engage in cryptocurrency mining, propagate malware, act as proxy networks for malicious activities, and enable remote control over compromised devices.

In the field of cybersecurity, a botnet functions as a covert group of compromised computers, commanded remotely by a malicious actor. These computers can collaborate to initiate disruptive activities, such as congesting websites with excessive traffic or proliferating malware and unsolicited emails. The pressing issue arises due to the fact that these malicious agents can manipulate numerous systems surreptitiously. Botnets are able to infiltrate a range of devices, from standard computers to servers and smart devices, resulting in a widespread and challenging threat. Botnet operations typically unfold in several distinct steps: propagation, infection, Command and Control (C&C) communication, and the execution of attacks. Examples of notable IoT botnets include Mirai and Bashlite.

The Mirai botnet that has emerged in 2016, can be given as a strong example of why IoT networks have so much potential for malicious actors. Mirai botnet has the capacity to generate network traffic of approximately up to 600 GB per second. It is notable that Mirai botnet did not target Windows devices, but rather IoT devices with Linux that are possibly has so many vulnerabilities (Idriss, 2020).

### 2.2.1 Architecture

The controlling mechanism can vary in terms of client-server architectures. Basically, there are two types of botnet architectures: centralized and decentralized (Dhayal and Kumar, 2018).

#### 2.2.1.1 Centralized

Centralized architecture is a basic client-server model. Botmaster acts like a server, and bots act like clients. This brings a serious vulnerability with it. Centralized C2 is where all connections between botmaster and bots are provided, and if it is eliminated or disabled, the entire botnet system would be taken down from a single-point failure.

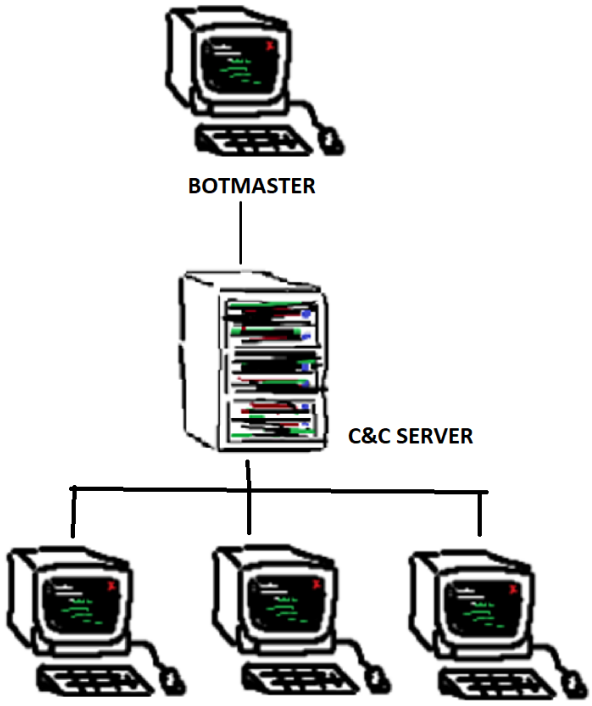


**Figure 2.1 :** Centralized botnet architecture.

#### 2.2.1.2 Decentralized

To avoid the single point of failure, bots can behave like a server, where taking down the botmaster would not take down the botnet. It is also more sophisticated to determine the connections, since the botmaster does not require to establish a connection with all bots in the botnet.

### 2.3 IoT Security

The Internet of Things (IoT) has experienced exponential growth, connecting diverse physical devices equipped with sensors and communication capabilities to collect and exchange data online. This proliferation, while enabling innovation across various sectors like smart homes, healthcare, and transportation, also significantly expands the attack surface for cyber threats, particularly botnets. IoT devices present unique

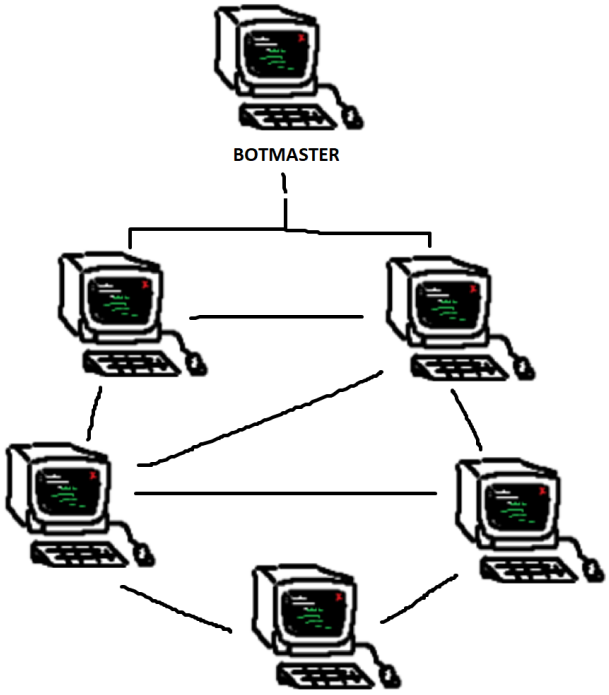


**Figure 2.2 :** Decentralized botnet architecture.

security challenges compared to traditional computing systems. These challenges include constraint of resources. And they are vulnerable generally.

## 2.4 Federated Learning

Federated Learning (FL) is a recent advancement in distributed machine learning that enables models to be trained across multiple decentralized devices or clients while keeping the raw data localized. This decentralized paradigm offers a compelling solution for scenarios involving sensitive data spread across numerous devices, such as in IoT networks. The standard FL process typically involves local training where each selected client trains the model locally using its own dataset. And using model aggregation phase, it will be possible to collect parameters and aggregate them with an algorithm -usually FedAvg is preferred- and update the global model.

## 3. LITERATURE REVIEW

Machine learning techniques have been extensively researched for IDS and detecting malicious activities, including botnets. ML-based IDS can distinguish between normal and abnormal network traffic patterns. Various ML algorithms have been applied, including ensemble methods like Random Forest and Decision Trees , k-Nearest Neighbors, and deep learning models such as Autoencoders and Convolutional Neural Networks (Rathod et al., 2024).

Federated Learning is a decentralized machine learning structure that trains models across multiple clients that have their own data in a local environment, without exchanging the raw data. This makes the system privacy-preserving. It is well-suited for scenarios with distributed data, such as in IoT networks. In FL, clients collaboratively train a global model by sharing model updates like weights with the central server, which aggregates these updates by using some aggregation algorithms, primarily FedAvg. The term called training rounds is used in FL to behave like Random Forest, where it selects features randomly, and FL chooses clients randomly for each training round. To enhance the concept and maintain the consistency and heterogeneity, there are client selection methods called client prioritizing (Fu et al., 2023).

A significant challenge in training machine learning models is data labeling, particularly if the data is network data, where the data count is really high for manual labeling. Since new attacks emerge frequently, manual labeling is an intensive process, and it consumes time. This is where generating synthetic data can help to address this issue (Randhawa et al., 2021).

GANs are one of the powerful generative models. They are capable of accurately modeling complex data distributions and generating synthetic data, such as malware and even network traffic (Devadiga et al., 2023). GANs can be used to generate synthetic network packets or traffic flows (Nukavarapu et al., 2022). In cybersecurity research, synthetic data generated by GANs can be used to augment training data for

intrusion detection systems and potentially, to generate realistic malware traffic to build robustness for defensive mechanisms in mobile devices (Dayan et al., 2023). GANs are also used for hardening for IDS systems in a study that builds a combination with AI (Randhawa et al., 2021).

The N-BaIoT dataset is a widely used public dataset specifically designed to detect IoT botnet attacks. It addresses the lack of publicly available datasets featuring real traffic from compromised IoT devices. The dataset contains real network traffic data collected from nine different commercial IoT devices that were infected with authentic Mirai and Bashlite botnets. A research on comparing different ML models on the same dataset shows that tree-based models, especially Random Forest, have significant performance (Hossain et al., 2024). In another study, several feature selection methods were performed on the N-BaIoT dataset, and PCA methods were found to be robust for detection with autoencoders (Sakthipriya et al., 2023).

**Table 3.1 :** Studies using N-BaIoT dataset.

| Ref | Feature Selection | Model | Accuracy | Synthetic |
|---|---|---|---|---|
| [14] | 15 Features | Random Forest (compares 6 different ML models) | 0.92 | SMOTE |
| [16] | 10 Features (MI and H for 5 tw) | Decision Tree (J48) | 0.99495 | Non |
| [17] | Autoencoder Internal Feature Reduction | Deep Autoencoder | 100% TPR and 0.07% FPR | Non |
| [18] | Yes | Random Forest vs XGBoost | 0.9918 XGBoost and 0.9921 Random Forest | Non |
| [19] | 5 Features (HH_jit_mean for 5 tw) | Random Tree, Random Forest and Random Committee | 0.9599 on a different dataset, starts from 0.9910 | Non |
| [20] | PCA | Random Forest, XGBoost RNN and ANN | 0.9990 | SMOTE |

To enhance the security of IoT devices, recently many machine learning algorithms explored for botnet detection purposes. For instance, Rawat et al. (2024) developed an algorithm to get a model that consists of several machine learning models such as Random Forest, XGBoost, ANN and RNN. With this approach, a final neural network was used to classify the botnet type in the N-BaIoT dataset. The developed model has 0.9990 accuracy for the test data. While they implement the SMOTE technique for data balancing, they used it only once before the training process. Also, they have mentioned federated learning methods as a future work, especially to solve the issues that need heavy computation and privacy issues on sensitive data.

In a related study, six different machine learning models were evaluated to determine their performance in a classification task. The findings of the study indicated that the Random Forest model demonstrated the highest level of accuracy among the models that were evaluated. It is noteworthy that the study utilised the SMOTE technique in the initial stage to address class imbalance, a factor that likely contributed to the enhanced performance. The mean accuracy of the Random Forest model across nine devices was 0.92, as determined through calculation. This establishes a useful benchmark for comparison with our own study, particularly with regard to the impact of synthetic data generation, both in terms of its initial generation and its subsequent development (Hossain et al., 2024).

# 4. FEDERATED LEARNING FRAMEWORK FOR IDS SYSTEMS

## 4.1 N-BaIoT Dataset

In this research, we have utilized the N-BaIoT dataset provided by UCI or Kaggle [22]. The N-BaIoT dataset contains statistical features extracted from network traffic collected from nine different IoT devices. Raw network traffic in IoT devices contains benign traffic as well, alongside malicious traffic. Some of these devices are infected with the Mirai and Bashlite botnets. Some of the IoT devices are infected with both botnets, while others are infected with only the Mirai botnet. This distribution of data makes the dataset Non Identical and Independent Distribution (non-IID), resulting in issues during the training phase, including the production of biased models. The resolution of these issues is possible through the implementation of a data balancing process, which involves the utilization of synthetic data generation techniques. These techniques will be integrated into the proposed model.

The device names and the botnets they are infected with are listed in Table 4.1. There are total of 9 different IoT devices. N-BaIoT is prepared by [23] for the Kitsune IDS, and they extracted 23 different statistical features while collecting the traffic and calculated for 5 different time windows as 100ms, 500ms, 1.5sec, 10sec, and 1min. Some features are 1D, meaning statistical features calculated from one stream only; on the other hand, there are also 2D features calculated between two streams [17].

**Table 4.1 :** Overview of IoT devices in N-BaIoT dataset.

| Device Name | Device Type | Infection |
|---|---|---|
| Danmini | Doorbell | Mirai & Bashlite |
| Ennio | Doorbell | Bashlite |
| Ecobee | Thermostat | Mirai & Bashlite |
| Philips B120N/10 | Baby Monitor | Mirai & Bashlite |
| Provision PT-737E | Security Camera | Mirai & Bashlite |
| Provision PT-838 | Security Camera | Mirai & Bashlite |
| SimpleHome XCS7-1002-WHT | Security Camera | Mirai & Bashlite |
| SimpleHome XCS7-1003-WHT | Security Camera | Mirai & Bashlite |
| Samsung SNH 1011 N | Webcam | Bashlite |

### 4.1.1 Preprocessing

The preprocessing of the data starts with cleaning the NaN values and removing duplicates. This action helps us to ensure that the N-BaIoT dataset is free from inconsistencies that can lead to skewed results. It also removes any redundant data. Following this, labels were encoded as 0 for benign samples and 1 for malicious samples since our DNN model requires encoded labels. Normalization is a crucial step to bring all values into a similar scale, especially when the dataset includes various statistical features with different ranges of values. This helps to improve the performance and convergence speed of many machine learning algorithms, including the DNN as our main model for detection. As a result of the cleaning process, the number of samples in our dataset was reduced to the following count shown in the Table 4.2.

**Table 4.2 :** Sample count by botnet type after dropping duplicates.

| | Benign | Mirai | Gafgyt | Total |
|---|---|---|---|---|
| #of Samples | 513,497 | 1,639,891 | 329,288 | 2,482,676 |

During the normalization step, we have employed StandarScaler from scikit-learn library which uses the equation (4.1) to standardize the data for non-biased decisions. This process transforms the values of each feature to have a mean of zero and a standard deviation of one. In this way, each feature contributes equally to the training process of the DNN model. While an improvement was observed in overall accuracy after the scaling process (Table 4.3), the actual benefit was the training time, which decreased nearly 90% (Table 4.4). The reason that prediction time is increased is that the data being predicted is exposed to a scaling process beforehand. The increase in prediction time remains minimal and does not pose a significant performance concern.

$$x' = \frac{x - \mu}{\sigma} \tag{4.1}$$

**Table 4.3 :** Performance comparison of DNN model with different feature sets.

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| No Scaling | 0.993307 | 0.993354 | 0.993307 | 0.993069 |
| Std Scaling | 0.999877 | 0.999877 | 0.999877 | 0.999877 |

**Table 4.4 :** Performance comparison of DNN model with different feature sets.

| Model | Time | |
|---|---|---|
| | Training Time | Prediction Time |
| No Scaling | 881.349394 | 0.000000 |
| Std Scaling | 88.104339 | 0.000726 |

## 4.2 Feature Selection

Feature selection plays a critical role in developing efficient machine learning models, particularly in resource-constrained environments such as those found in Internet of Things (IoT) applications. In such contexts, reducing computational complexity and memory usage is essential, and proper feature selection can also help mitigate overfitting. In the study presented in [24], the most relevant features were identified using a Random Forest model. Some studies [25] chose 19 features from the N-BaIoT dataset, where we have common features selected. This can give insight into the accuracy of our method. Initially, quasi-constant features that exhibit minimal variance were removed, as they contribute little predictive value. Subsequently, a correlation analysis was performed using a hybrid methodology that combined Gini importance and Mutual Information to evaluate and select features from highly correlated feature sets. As a result of this multi-step process, a total of ten features were selected. These selected features are listed in Table 4.5.

**Table 4.5 :** Selected feature names.

| Selected Features | |
|---|---|
| HH_jit_L0.1_weight | H_L1_variance |
| HH_L1_std | H_L3_weight |
| HH_jit_L0.01_mean | HpHp_L0.01_weight |
| MI_dir_L0.01_mean | HpHp_L3_std |
| H_L0.01_weight | HpHp_L5_weight |

### 4.2.1 Performance on neural networks

It is important to assess whether the selected features also perform well when used in neural network models. To this end, we evaluated the impact of feature selection on the performance of DNNs. Similar to a previous study conducted in the study [24], the feature selection process was carried out using a Random Forest model, and the evaluation metrics were initially calculated based on its output. In this study,

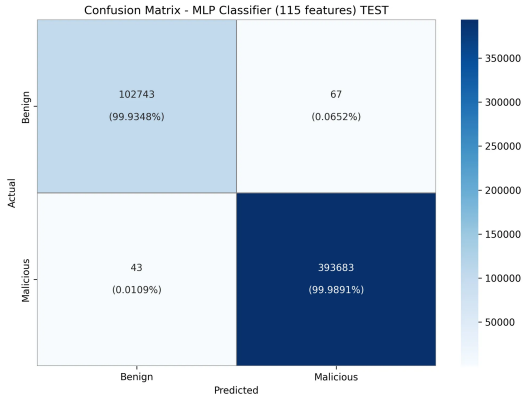


**Figure 4.1 :** Confusion matrix (115 features).

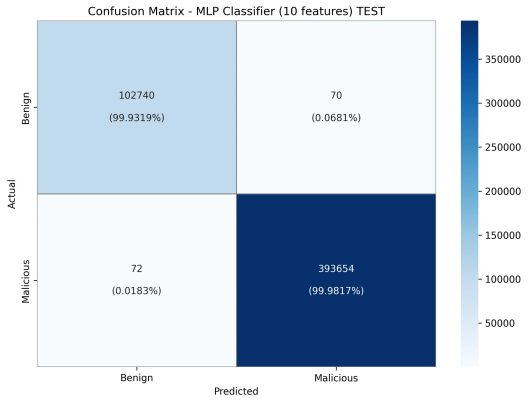


**Figure 4.2 :** Confusion matrix (10 features).

we applied the same selected features shown in Table 4.5 to a DNN and assessed performance using standard classification metrics: accuracy, precision, recall, and F1-score. As shown in Table 4.6, the performance of the DNN using the reduced feature set remained nearly the same compared to the full feature set, in addition to maintaining classification performance.

**Table 4.6 :** Performance comparison of DNN model with different feature sets.

| #of Features | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| 10 | 0.999714 | 0.999714 | 0.999714 | 0.999714 |
| 115 | 0.999778 | 0.999778 | 0.999778 | 0.999778 |

#### 4.2.1.1 Classification by label

Given that the number of false positives and the number of false negatives is approximately the same, if we compare the FP and FNs Figure4.1 and Figure4.2, it can be inferred that the selected features contribute to balanced and effective classification performance in terms of classifying the traffic as malicious and benign.

#### 4.2.1.2 Classification by botnet type

The model has low performance on Gafgyt botnet detection as seen in Figure 4.3 and Figure 4.4 . This may be the result of a low sample count or low feature quality while extracting the statistics from the network data.

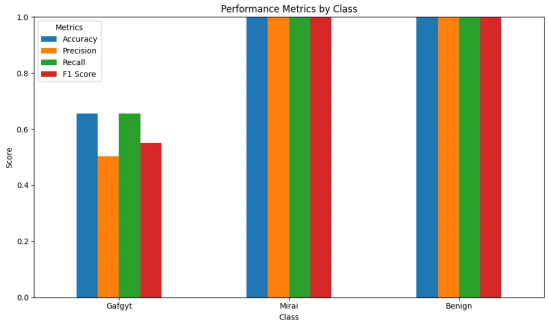


**Figure 4.3 :** Performance metrics - Botnet type (115 features).

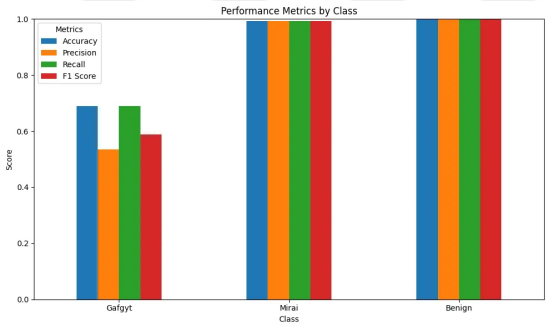


**Figure 4.4 :** Performance metrics - Botnet type (10 features).

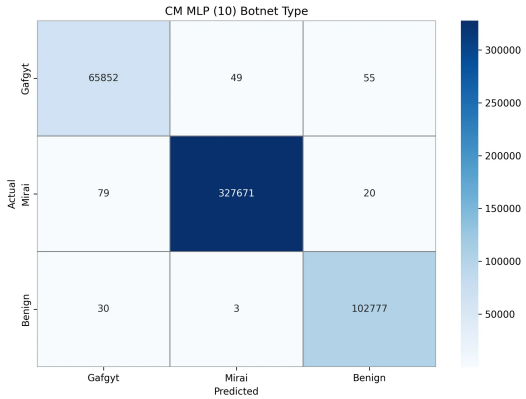


**Figure 4.5 :** Confusion matrix - Botnet type (10 features).

#### 4.2.1.3 Classification by attack code

The relatively poor classification performance of the Gafgyt attack appears to be a consequence of the model's inability to effectively differentiate between TCP and UDP traffic patterns. This finding suggests that the selected features may not effectively

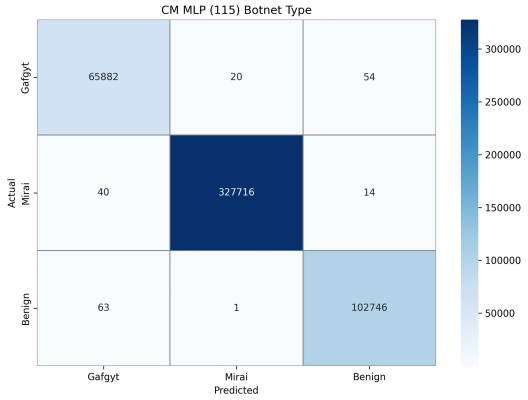


**Figure 4.6 :** Confusion matrix - Botnet type (115 features).

capture the protocol-specific characteristics necessary for an accurate identification of Gafgyt-based attacks. It is possible that the distinguishing attributes between TCP and UDP traffic, such as connection-oriented behaviour, session management, or packet structure, are either underrepresented or not sufficiently emphasised in the selected feature set. Further investigation into protocol-level feature representation may help improve classification performance for such attacks. You can see the details on Figure B.1 and Figure B.2

## 4.3 Synthetic Data Generation

### 4.3.1 CTGANs

Class imbalance is a common challenge in machine learning tasks, and it is essential to handle it with care to ensure the reliability of the model. Despite a variety of techniques, GANs have proven particular potential due to their strong learning mechanism. We have utilized Conditional Tabular Generative Adversarial Networks (CTGANs) since our data is tabular, and we need class balancing, where conditional GANs are actually good at their design to generate realistic data for a given class given as a condition.

Since the synthetic data will take place within IoT devices and we have already selected the top 10 features, synthetic data will also contain only 10 features.

### 4.3.2 Evaluation of synthetic data

The quality of the generated data is important in deciding whether we should rely on it or not. Some metrics will be used for the evaluation.

#### 4.3.2.1 Feature distribution comparison

We observe that the distributions of synthetic samples closely match the real ones, especially in terms of shape, spread, and skewness. This indicates CTGAN has successfully learned the underlying statistical properties of the minority class. You can look at the distribution comparison of selected features on **Appendix C**.

#### 4.3.2.2 Detection performance on DNN

**Table 4.7 :** Performance comparison of DNN model with and without the synthetic data.

| Data | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Real | 0.992373 | 0.991805 | 0.992373 | 0.991627 |
| Real + Synthetic | 0.993353 | 0.993395 | 0.993353 | 0.992472 |

### 4.3.3 Metric Comparison

We can also use some metrics to measure how realistic the synthetically generated data is. We have used Kolmogorov-Smirnov (KS) statistic, Wasserstein Distance and Kullback–Leibler (KL) divergence. You can see the results in Table A.3 A.1 A.2. In this way, we can understand which features are reliable as synthetic data.

## 4.4 Neural Networks

### 4.4.1 Hyperparameter tuning

Using sklearn GridSearchCV, we have decided on the best parameters: hidden layer sizes to be [64,32], and learning rate to be 0.01 as shown with the best accuracy in Table 4.8.

**Table 4.8 :** Hyperparameter tuning for DNN.

| Hidden Layers | LR | Accuracy |
|---|---|---|
| [64, 32] | 0.01 | **0.998925** |
| [64, 32] | 0.001 | 0.952291 |
| [32, 16] | 0.01 | 0.951795 |
| [32, 16] | 0.001 | 0.951911 |
| [32, 16, 8] | 0.01 | 0.951751 |
| [32, 16, 8] | 0.001 | 0.951752 |

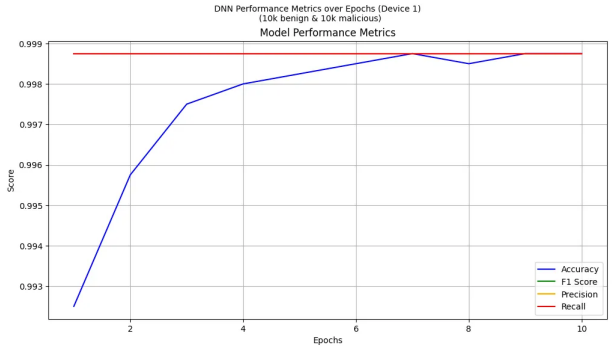


**Figure 4.7 :** DNN model performance on a balanced dataset over epochs.

### 4.4.2 Evaluation of DNN model

After finding the best parameters for the DNN model, we have trained the model with device-based data and evaluated the accuracy across the devices that can be seen in the Figure 4.8.

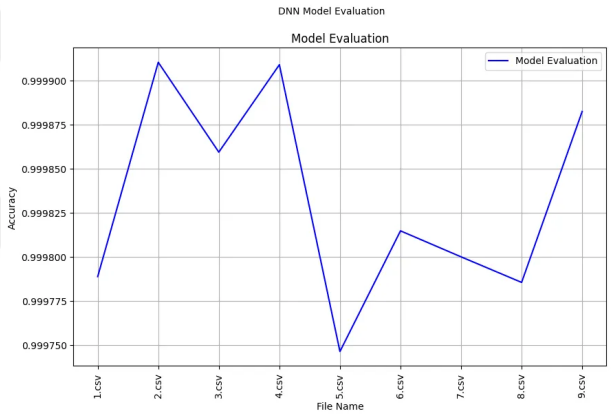


**Figure 4.8 :** Device-based accuracy of tuned DNN model.

## 4.5 Federated Learning

### 4.5.1 Proposed architecture

In federated learning, collaboration is the key. Since the IoT devices can contain various private data, collaborative learning should not be done with a central entity. Collecting data would violate the device's privacy. Instead of sending data to a central entity, sending only the model updates eliminates the data exchange between the client and the server. This results in the sensitive data to stay in the local device.

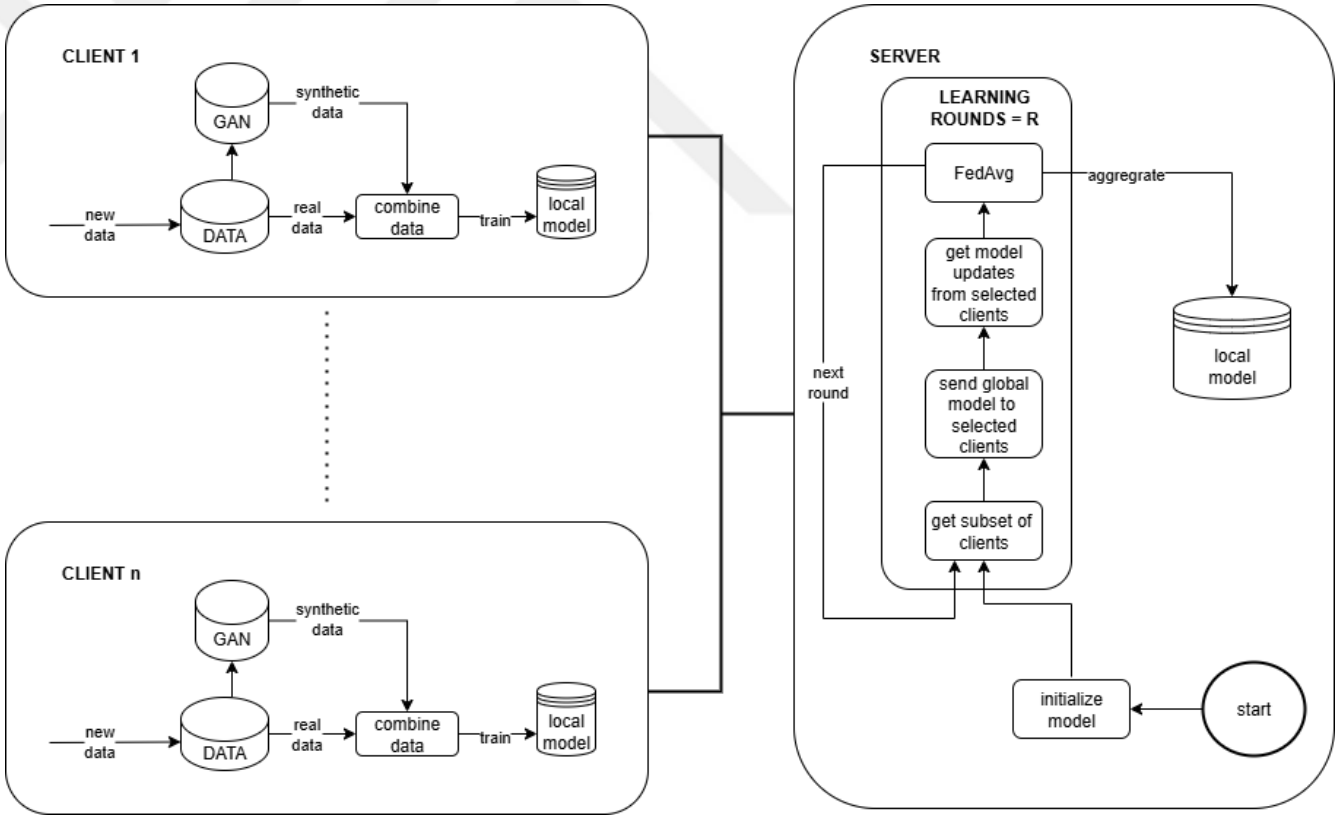


**Figure 4.9 :** Proposed architecture.

#### 4.5.1.1 Model Initialization

For the model initialization part, there are some studies done that collect public models' weights and decide which weights to start with for better performance. For simplicity, we have selected a random client to train a model with the selected client's local data, with the hyperparameters given by the server. Once the randomly selected client trains the model, it sends the coefficients and intercepts back to the server. This may lead to inconsistency of the global model, but since we have high accuracies in device-based models, this does not necessarily lead to severely biased results.

#### 4.5.1.2 Algorithm

After the server updates the global model with the model updates, it starts a training round for a decided number of rounds. Each round consists of these steps:

1. Select CLIENTS_PER_ROUND clients randomly
2. Send the global model to randomly selected clients
3. Randomly selected clients will generate synthetic data and combine it with the real data to train the model they fetch from the server.
4. After each round, randomly selected clients will send the model updates to the server
5. Server will use these updates to aggregate with the global model using FedAvg function.

### 4.5.2 Experiments

#### 4.5.2.1 Parameter optimization for federated learning

To determine the optimal values, we experimented with various combinations of two variables: training round (R) and client per round (C). Specifically, for the number of training rounds, we used values of 3, 5, and 10. For clients per round, we used values of 2, 3, 4, and 5, resulting in a total of 12 different federated learning configurations. As both the initial model's client selection and the clients chosen in each round are random, each configuration was repeated 10 times, and the results were averaged. Each full repetition of the federated learning process is referred to as a *rotation*. This strategy was adopted to reduce the bias caused by random client selection, thereby ensuring more stable and reliable outcomes. As the graphs show, it is important to distinguish

between the concepts of *rotation* and *round*. While a rotation refers to one iteration of the federated training process, a round is each federated learning step that the server waits for. At the end of the repeated federated learning process, which is a total of 10 rotations, the detection accuracy was computed by averaging the results of the final global model, which was trained collaboratively across clients and evaluated on each participating client's device-specific test data.

In this way, we have measured the effects of the number of training rounds and client-per-round variables. The average accuracy across devices was measured as highest with 5 training rounds and 3 clients per round. It can be seen in the Table 4.9.

**Table 4.9 :** Best parameter for training round (R), client per round (C), and epoch (10).

| (R,C,E) | Accuracy (Avg.) |
|---|---|
| (3, 2, 10) | 0.999585 |
| (3, 3, 10) | 0.999583 |
| (3, 4, 10) | 0.999570 |
| (3, 5, 10) | 0.999583 |
| (5, 2, 10) | 0.999585 |
| (5, 3, 10) | **0.999628** |
| (5, 4, 10) | 0.999590 |
| (5, 5, 10) | 0.999565 |
| (10, 2, 10) | 0.999600 |
| (10, 3, 10) | 0.999585 |
| (10, 4, 10) | 0.999589 |
| (10, 5, 10) | 0.999600 |

#### 4.5.2.2 Synthetic data generation integration point

Three experiments were conducted in order to determine the effect of the inclusion of a synthetic data generation process at a specific stage of the federated learning process. This was achieved by means of a review of the literature, with a view to identifying the manner in which other studies had previously approached the subject. In the initial iteration, synthetic data generation was not incorporated. InFed presents the results obtained by incorporating it into the federated learning process, as outlined in the proposed model. OutFed demonstrates the results obtained by first performing data balancing and subsequently training the model, as is the case in numerous studies in the literature. In our case, OutFed, synthetic data is produced prior to the initiation of federated processes, and CTGAN is not included in the federated learning process.

**NoSyn:**

In this experiment we do not include CTGAN into the proposed architecture as in Figure 4.11. The only difference is not using CTGAN in any stage. The same measurements were done using the defined values of training round as 5 and client per round as 3. The avarage of the accuracies was calculated after repeating the federated learning process 10 times as it was before. All experiments will be done with 10 rotations and then averaging the results.

**InFed:**

This experiment is the exact implementation of the proposed architecture previously shown in Figure 4.9. The synthetic data generation process is integrated into the federated learning process to continuously balancing the data inside the IoT device to get better results. The outcomes have been ascertained through the calculation of the mean of ten rotations, thus ensuring the elimination of randomness within the federated learning process.

**OutFed:**

Synthetic data generation placed outside the federated learning process as a preliminary step before initiating the federated learning process. Each device generates its own synthetic data using CTGANs to balance their local data where it can not solve the future unbalance data issues since that is used one time as a preliminary step. The synthetic data generation step is first carried out as it shown in Figure 4.10. Then the same steps followed as in the NoSyn architecture Figure 4.11.

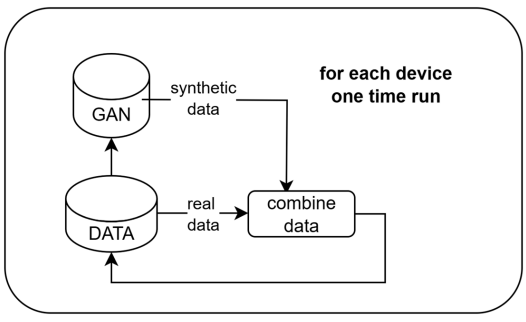


**Figure 4.10 :** Preliminary step for OutFed architecture.

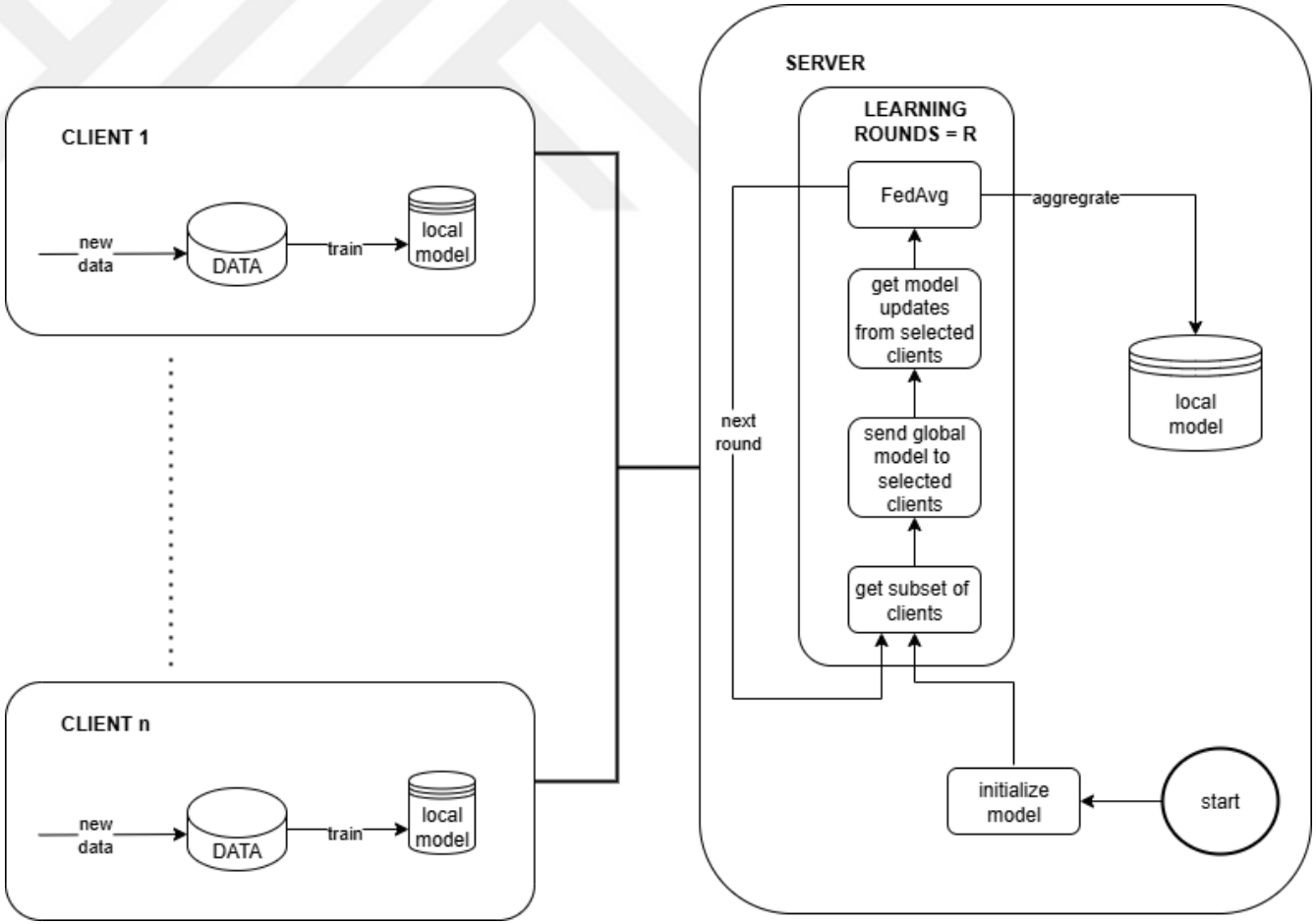


**Figure 4.11 :** NoSyn architecture.

### 4.5.3 Results

In this section, the results obtained from the implementation of the experiments mentioned in the section 4.5.2.2 along with their diagrams, will be presented. These experiments involved 10 rotations each. In Figure 4.12, 4.13, and 4.14, the total time of a federated learning process is shown for each rotation. As in NoSyn and OutFed experiments, we can see that the average time is similar, which is nearly 480 milliseconds. On the other hand, InFed architecture needs more time for each rotation because the synthetic data generation process is included for each round. As can be seen in Figure 4.13, the time needed for a single rotation is 1140 milliseconds on average. This result is acceptable for a single federated learning process.

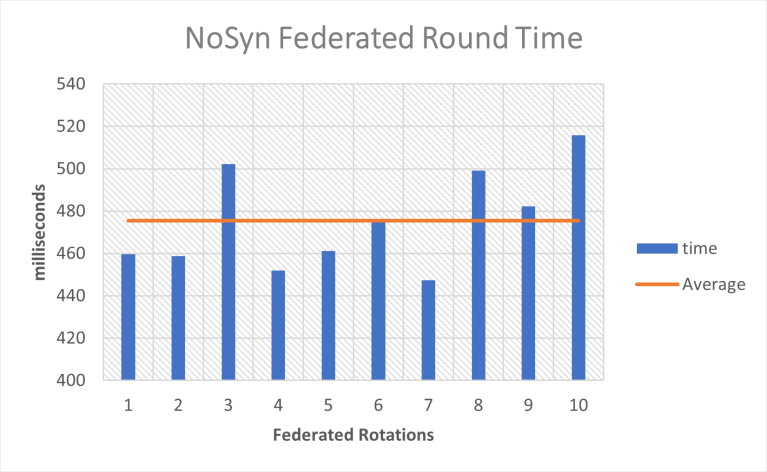


**Figure 4.12 :** NoSyn experiment runtime for 10 rotations.

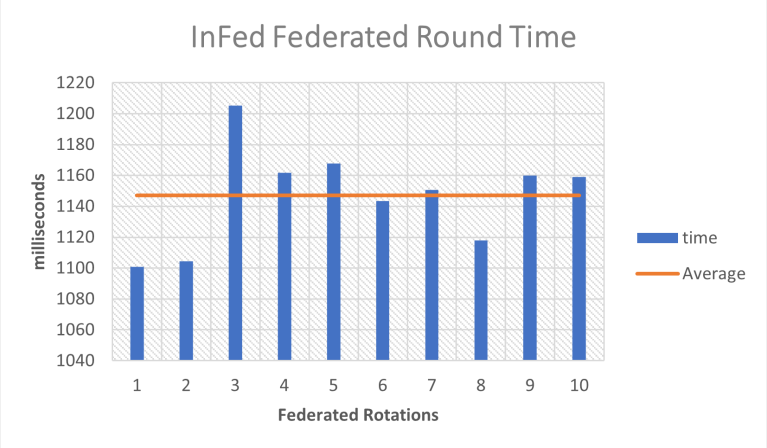


**Figure 4.13 :** InFed experiment runtime for 10 rotations.

Each federated learning experiment was repeated for 10 rotations, and the accuracy of each round was averaged across those rotations. As demonstrated in the graph shown in Figure 4.15, the NoSyn experiment demonstrates low accuracy at the beginning and

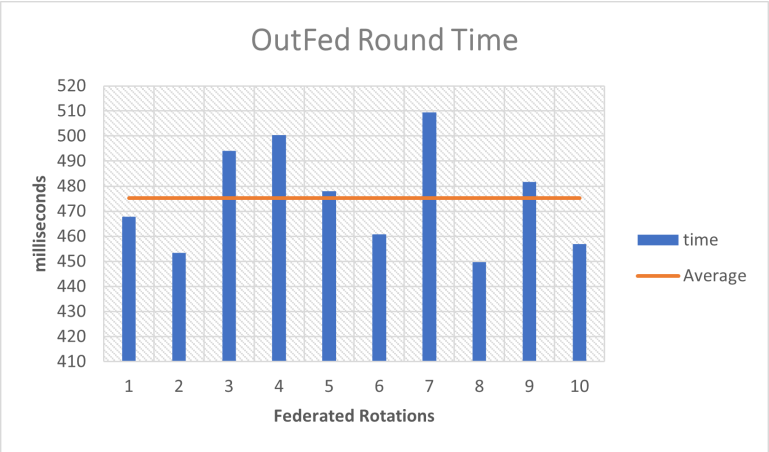


**Figure 4.14 :** OutFed experiment runtime for 10 rotations.

produces lower performance compared to the other experiments. A comparison of the InFed and OutFed experiments reveals that starting with a balanced dataset has a positive effect, as evidenced by the results of the OutFed experiment. However, no significant increase in accuracy is observed in the OutFed experiment. In contrast, in the InFed experiment, a continuous and consistent increase in accuracy is observed with the addition of synthetic data generation at each round.

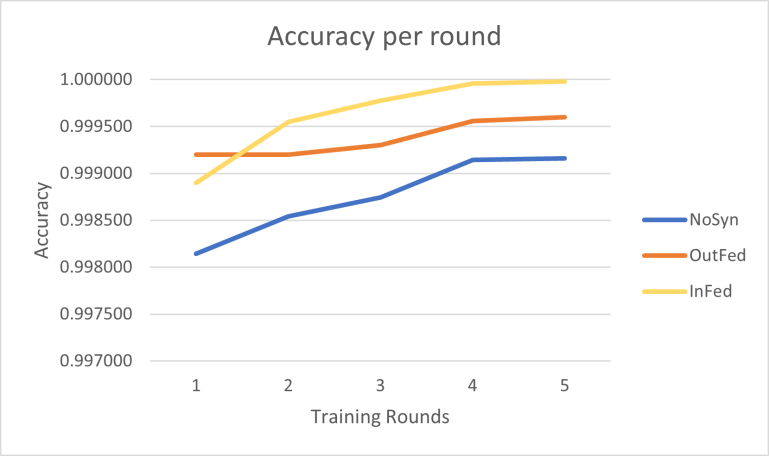


**Figure 4.15 :** Average accuracy for each experiment NoSyn, InFed, and OutFed.

Consequently, the proposed architecture, evaluated in the InFed experiment, demonstrates that integrating synthetic data generation into the federated learning process yields better performance in comparison to utilizing it only as a preliminary step or not using it at all. As a result, with 5 training rounds and using synthetic generation for each step by integrating it into the federated learning process, we have reached an accuracy value of 0.9999 using our DNN model with the proposed architecture. On the other hand, Rawat et al. (2024) proposed a hybrid machine

learning approach by combining 4 different machine learning models, such as Random Forest, XGBoost, ANN, and RNN, to classify whether the traffic is a botnet traffic or benign traffic in the N-BaIoT dataset, achieving a test accuracy of 0.9990. Despite the implementation of the SMOTE technique for data balancing, it was utilized prior to the training phase as a preliminary step. Additionally, their work did not incorporate federated learning, but they identified a future research direction. They are also addressing privacy and computational concerns through the integration of federated learning. Even our OutFed experiment resulted better than the mentioned study, as we have reached above 0.9995 by generating synthetic data only in the first place as prior to the federated learning process.

# 5. CONCLUSION

Effective botnet detection has a huge importance when it comes to IoT networks. Botnet threats can demonstrate attacks with huge impacts, like DDoS attacks. Since IoT networks hold strong potential despite their low resources, their connectivity and vulnerabilities can serve botnets very effectively. There are several machine learning models that are preferred by IDS solutions. However, federated learning, which requires model aggregation on the server side, has more appropriate machine learning models, such as NN-based models that have a more suitable nature for the aggregation mechanisms. FedAvg is a commonly used and well-performing aggregation function. This function takes the model updates and aggregates with the global model by averaging all *coefs_* and *intercepts_* received from the clients.

Resource limitation on IoT devices requires dimensionality reduction if detection is to be demonstrated on the resource-limited device using a huge amount of data. Dimensionality reduction was done by selecting the most important features with high predictive power using a hybrid selection method. This method consists of several steps: applying a variance threshold for elimination of features with low variance as low predictive power on the target, correlation analysis of features and calculating Gini importance using Random Forest model and mutual information to find the best features among the correlated features. In this way, 10 features are selected from 115 features in total.

Our DNN model performs very well event with the selected 10 features from 115 features. Accuracy, precision, recall and F1 scores shows similar results for both cases where DNN trained with 115 features versus 10 features. This shows that we can only use these features from the data.

Classification results compared against botnet type, attack code and label. We can see that gafgyt botnet has low performance since the *udp_tcp* and *udp_tcp* attacks can not differed each other. Also with small number of samples. Benign traffic is also low sample count.

The 2,482,676 samples for 9 IoT devices from N-BaIoT dataset is used to train local models on IoT devices. Since the dataset is imbalanced. We have utilized the CTGANs to balance dataset for better performance. Before using the synthetic data is evaluated with 5 different methods. We have manually checked the distributions of each feature and saw the resemblance. KS Statistics, KL Divergence and Wasserstein Distance to statistically compare the distributions even further. The results shows that CTGANs generates realistic data for 3 features which are *HpHp_L3_std*, *HpHp_L5_weight* and *HH_L1_std*. We also trained DNN model with real data and combined with synthetic data to see the performance results where the usage of the synthetic data increased accuracy and other metrics.

After hyper parameter tuning in DNN model we have used 2 hidden layer with sizes 64, 32 respectively and 0.01 as learning rate which give the best accuracy for other combinations as you can see in the Table 4.8. This DNN model converges after 10 epochs why we use 10 epochs afterwards. But this model does not demonstrate the same performance across the devices. Since we have non-IID data on our devices it is normal to see this results like in Figure 4.7.

We have evaluated the proposed federated architecture with some parameters. For 5 training rounds, 3 clients per round and 10 epoch resulted as the best accuracy. To evaluate our architecture and observe the effect of the integration point of the synthetic data generation process. We have carried out 3 different experiment for this. At first we do not involve any synthetic data generation process as in Figure 4.11. As seen in Figure 4.15 exclusion of the synthetic data generation process leads low accuracies in the begining of the federated learning process and does not increased the accuracy as much as InFed architecture. On the other hand OutFed shows a good start since the dataset is balanced before the federated learning process is started. However as new traffic comming there is no further data balancing in the OutFed achitecture. This leads a normal increase in the accuracy along with the training rounds. As the best performer in these architectures, our proposed model InFed resulted better then others with the synthetic data generation process integrated in the federated learning process.

As a conclusion synthetic data generation is important and useful when it comes to unbalanced data becomes a problem. Especially when the synthetic data generation is integrated into the federated learning process we see better results. Therefore

integration a GAN to the federated learning system was helpful to improve the detection rates. We get 0.9999 detection accuracy for the proposed architecture on N-BaIoT dataset which outperforms against the other related studies that utilize N-BaIoT dataset as can be seen in the Table3.1.

## REFERENCES


[1] **Suchetha, G. and Pushpalatha, K.** (2024). Optimizing botnet detection in iot networks: Feature selection analysis on the unsw-nb15 dataset, *8th IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics, DISCOVER 2024 - Proceedings*, pp. 120–125, doi : 10.1109/DISCOVER62353.2024.10750583.

[2] **Hamid, H.**, **Noor, R.M.**, **Omar, S.N.**, **Ahmedy, I.**, **Anjum, S.S.**, **Shah, S.A.A.**, **Kaur, S.**, **Othman, F.**, **and Tamil, E.M.** (2021). Iot-based botnet attacks systematic mapping study of literature, *Scientometrics*, *126*, 2759–2800, doi : 10.1007/S11192-020-03819-5/TABLES/14. `https://link.springer.com/article/10.1007/s11192-020-03819-5`.

[3] **Rathod, G.**, **Sabnis, V.**, **and Jain, J.K.** (2024). Improving iot botnet attack detection using machine learning: Comparative analysis of feature selection methods and classifiers in intrusion detection systems, *In 2024 3rd International Conference for Innovation in Technology (INOCON)*, pp. 1–8. IEEE, doi : 10.1109/INOCON60754.2024.10511883. `https://ieeexplore.ieee.org/document/10511883/`.

[4] **Saveetha, D.**, **Maragatham, G.**, **Ponnusamy, V.**, **and Zdravkovic, N.** (2024). An integrated federated machine learning and blockchain framework with optimal miner selection for reliable ddos attack detection, *IEEE Access*, doi : 10.1109/ACCESS.2024.3413076.

[5] **Lim, P.A.E. and Park, C.H.** (2024). A collaborative ensemble construction method for federated random forest, *Expert Systems with Applications*, *255*, 124742, doi : 10.1016/J.ESWA.2024.124742. `https://www.sciencedirect.com/science/article/pii/S0957417424016099`.

[6] **Popoola, S.I.**, **Ande, R.**, **Adebisi, B.**, **Gui, G.**, **Hammoudeh, M.**, **and Jogunola, O.** (2022). Federated deep learning for zero-day botnet attack detection in iot-edge devices, *IEEE Internet of Things Journal*, *9*(5), 3930–3944, doi : 10.1109/JIOT.2021.3100755.

[7] **Idriss, H.k.** (2020). Mirai botnet in lebanon, *In 2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–6, doi : 10.1109/ISDFS49300.2020.9116456.

[8] **Dhayal, H. and Kumar, J.** (2018). Botnet and p2p botnet detection strategies: A review, *In 2018 International Conference on Communication and Signal Processing (ICCSP)*, pp. 1077–1082, doi : 10.1109/ICCSP.2018.8524529.

[9] **Fu, L.**, **Zhang, H.**, **Gao, G.**, **Zhang, M.**, **and Liu, X.** (2023). Client selection in federated learning: Principles, challenges, and opportunities, *IEEE Internet of Things Journal*, *10*, 21811–21819, doi : 10.1109/JIOT.2023.3299573.

[10] **Randhawa, R.H.**, **Aslam, N.**, **Alauthman, M.**, **Rafiq, H.**, **and Comeau, F.** (2021). Security hardening of botnet detectors using generative adversarial networks, *IEEE Access*, *9*, 78276–78292, doi : 10.1109/ACCESS.2021.3083421.

[11] **Devadiga, D.**, **Koo, H.**, **Singh, A.**, **Jin, G.**, **Han, A.**, **Chaudhari, K.**, **Potdar, B.**, **Shringi, A.**, **and Kumar, S.** (2023). Gleam: Gan and llm for evasive adversarial malware, *International Conference on ICT Convergence*, pp. 53–58, doi : 10.1109/ICTC58733.2023.10393706.

[12] **Nukavarapu, S.K.**, **Ayyat, M.**, **and Nadeem, T.** (2022). Miragenet - towards a gan-based framework for synthetic network traffic generation, *Proceedings - IEEE Global Communications Conference, GLOBECOM*, pp. 3089–3095, doi : 10.1109/GLOBECOM48099.2022.10001494.

[13] **Dayan, O.**, **Wolf, L.**, **Wang, F.**, **and Harel, Y.** (2023). Optimizing ai for mobile malware detection by self-built-dataset gan oversampling and lgbm, *Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience, CSR 2023*, pp. 60–65, doi : 10.1109/CSR57506.2023.10224927.

[14] **Hossain, M.**, **Rudro, R.A.M.**, **Razzaque, R.**, **and Nur, K.** (2024). Machine learning approaches for detecting iot botnet attacks: A comparative study of n-baiot dataset, *In 2024 International Conference on Decision Aid Sciences and Applications (DASA)*, pp. 1–7. IEEE, doi : 10.1109/DASA63652.2024.10836622. `https://ieeexplore.ieee.org/document/10836622/`, EFERİM :).

[15] **Sakthipriya, N.**, **Govindasamy, V.**, **and Akila, V.** (2023). A comparative analysis of various dimensionality reduction techniques on n-baiot dataset for iot botnet detection, *In 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*, pp. 1–6. IEEE, doi : 10.1109/PCEMS58491.2023.10136065. `https://ieeexplore.ieee.org/document/10136065/`.

[16] **OKUR, C. and DENER, M.** (2020). Detecting iot botnet attacks using machine learning methods, *In 2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, pp. 31–37, doi : 10.1109/ISCTURKEY51113.2020.9307994.

[17] **Meidan, Y.**, **Bohadana, M.**, **Mathov, Y.**, **Mirsky, Y.**, **Breitenbacher, D.**, **Shabtai, A.**, **and Elovici, Y.** (202x). N-baiot: Network-based detection of iot botnet attacks using deep autoencoders, *IEEE PERVASIVE COMPUTING*, *13*. `http://archive.ics.uci.edu/ml/datasets/detection`.

[18] **Zhukabayeva, T.**, **Zholshiyeva, L.**, **Ven-Tsen, K.**, **Adamova, A.**, **Mardenov, Y.**, **and Karabayev, N.** (2024). Enhancing iot security: Effective botnet attack detection through machine learning, *Procedia Computer Science*, *241*, 421–426, doi : https://doi.org/10.1016/j.procs.2024.08.058. https://www.sciencedirect.com/science/article/pii/S1877050924017691, 19th International Conference on Future Networks and Communications/ 21th International Conference on Mobile Systems and Pervasive Computing/14th International Conference on Sustainable Energy Information Technology.

[19] **Natarajan, D.**, **Katiravan, J.**, **and S.P, S.** (2024). Botnet attack detection in iot devices using ensemble classifiers with reduced feature space, *International Research Journal of Multidisciplinary Technovation*, pp. 274–295, doi : 10.54392/irjmt24321.

[20] **Rawat, M.**, **Bedi, A.S.**, **Singh, B.**, **Gupta, S.**, **Singal, G.**, **and Kaur, P.** (2024). Ensemble-based botnet attack detection and classification using machine learning algorithms on nbaiot dataset, *In 2024 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6. IEEE, doi : 10.1109/TENSYMP61132.2024.10752221. https://ieeexplore.ieee.org/document/10752221/.

[21] **Rawat, M.**, **Singh Bedi, A.**, **Singh, B.**, **Gupta, S.**, **Singal, G.**, **and Kaur, P.** (2024). Ensemble-based botnet attack detection and classification using machine learning algorithms on nbaiot dataset, *In 2024 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6, doi : 10.1109/TENSYMP61132.2024.10752221.

[22] **URL-1**. <https://archive.ics.uci.edu/dataset/442/detection+of+iot+botnet+attacks+n+baiot>, date retrieved: 28.05.2025.

[23] **Mirsky, Y.**, **Doitshman, T.**, **Elovici, Y.**, **and Shabtai, A.** (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection, *In Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, doi : 10.14722/ndss.2018.23204. https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-3_Mirsky_paper.pdf.

[24] **Uslan, B.** (2025). Enhancing iot botnet detection: A streamlined approach to feature selection, *International Graduate Research Symposium (IGRS)*.

[25] **Abbasi, F.**, **Naderan, M.**, **and Alavi, S.E.** (2021). Anomaly detection in internet of things using feature selection and classification based on logistic regression and artificial neural network on n-baiot dataset, *In 2021 5th International Conference on Internet of Things and Applications (IoT)*, pp. 1–7. IEEE, doi : 10.1109/IoT52625.2021.9469605. https://ieeexplore.ieee.org/document/9469605/.

**APPENDIX**

**APPENDIX A :** Synthetic Data Evaluation Metrics
**APPENDIX B :** Confusion Matrices
**APPENDIX C :** Feature Distributions

## APPENDIX A

Synthetic data is generated by CTGANs where all devices has one fitted with the local data. This will balance the dataset in each device. Three different metrics are used to evaluate the synthetic data as you can see in TableA.1 , TableA.2 and TableA.3. Bold values can be seen as realistic data. KL Divergence and Wassertein Distance metrics shows that ***_std*** features generated realistically in general among the devices.

**Table A.1 :** KS Statistics results of each feature from synthetic data by device id.

| Metric | Features | Device ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 |
| KS Statistics | HH_jit_L0.1_weight | 0.2823 | 0.3404 | 0.4151 | 0.2310 | 0.3444 | 0.3136 | 0.4642 | 0.2824 | 0.3005 |
| | HH_L1_std | 0.5414 | 0.3213 | 0.5870 | 0.3375 | 0.4536 | 0.5168 | 0.4781 | 0.2830 | 0.3455 |
| | HH_jit_L0.01_mean | 0.3653 | 0.3515 | 0.4737 | 0.3446 | 0.3869 | 0.3863 | 0.5271 | 0.2967 | 0.3451 |
| | MI_dir_L0.01_mean | 0.2070 | 0.1888 | 0.4084 | 0.2338 | 0.2772 | 0.3212 | 0.2885 | 0.1828 | 0.2411 |
| | H_L0.01_weight | 0.3157 | 0.2909 | 0.2859 | 0.2966 | 0.2942 | 0.3111 | 0.4067 | 0.3145 | 0.2971 |
| | H_L1_variance | 0.2735 | 0.1943 | 0.3262 | 0.3619 | 0.3163 | 0.2736 | 0.3987 | 0.2077 | 0.1760 |
| | H_L3_weight | 0.2601 | 0.2478 | 0.4059 | 0.2571 | 0.3514 | 0.2791 | 0.4111 | 0.2787 | 0.2520 |
| | HpHp_L0.01_weight | 0.5959 | 0.6938 | 0.4143 | 0.7535 | 0.4239 | 0.7219 | 0.4084 | 0.4466 | 0.4874 |
| | HpHp_L3_std | 0.5854 | 0.6737 | 0.5105 | 0.6548 | 0.5034 | 0.5234 | 0.5654 | 0.5951 | 0.5002 |
| | HpHp_L5_weight | 0.8306 | 0.7155 | 0.4938 | 0.5374 | 0.6373 | 0.7767 | 0.7964 | 0.4763 | 0.6679 |

**Table A.2 :** Wassertein Distance results of each feature from synthetic data by device id.

| Metric | Features | Device ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 |
| Wassertein Distance | HH_jit_L0.1_weight | 0.3060 | 0.4045 | 0.3141 | 0.2287 | 0.2190 | 0.2604 | 0.3220 | 0.3734 | 0.4391 |
| | HH_L1_std | **0.0761** | 0.3762 | **0.0807** | **0.0994** | 0.6128 | 0.5910 | 0.2541 | **0.0854** | **0.0675** |
| | HH_jit_L0.01_mean | 0.4679 | 0.3870 | 0.8514 | 0.3914 | 0.6253 | 0.5978 | 0.6948 | 0.4332 | 0.3461 |
| | MI_dir_L0.01_mean | 0.2533 | 0.1569 | 0.8157 | 0.3696 | 0.4394 | 0.3739 | 0.7893 | 0.1814 | 0.1951 |
| | H_L0.01_weight | 0.4297 | 0.3388 | 0.2770 | 0.4105 | 0.4259 | 0.4184 | 0.2659 | 0.4488 | 0.3727 |
| | H_L1_variance | 0.5458 | 0.2566 | **0.0658** | 0.4353 | 0.2033 | 0.1646 | 0.2403 | 0.3955 | 0.2057 |
| | H_L3_weight | 0.5211 | 0.4714 | 0.3345 | 0.4718 | 0.5771 | 0.4064 | 0.3882 | 0.5426 | 0.4389 |
| | HpHp_L0.01_weight | 0.1430 | 0.1535 | **0.1068** | 0.1106 | 0.1199 | **0.0878** | 0.1745 | 0.1022 | 0.1565 |
| | HpHp_L3_std | **0.0077** | **0.0746** | **0.0252** | **0.0343** | 0.1001 | 0.1891 | 0.1706 | **0.0208** | **0.0125** |
| | HpHp_L5_weight | **0.0819** | **0.0776** | **0.0187** | **0.0636** | **0.0741** | 0.0640 | 0.1958 | **0.0529** | 0.1389 |

**Table A.3 :** KL Divergence results of each feature from synthetic data by device id.

| Metric | Features | Device ID | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 |
| KL Divergence | HH_jit_L0.1_weight | 4.0990 | 2.3406 | 2.3364 | 2.8264 | 1.9885 | 2.9349 | 1.8286 | 2.1998 | 2.7014 |
| | HH_L1_std | **0.0077** | 0.3709 | **0.0887** | **0.1404** | 0.4909 | 1.4792 | 0.1282 | **0.0333** | **0.0038** |
| | HH_jit_L0.01_mean | 1.8524 | 2.1866 | 9.4444 | 1.5566 | 3.9393 | 2.6131 | 6.4092 | 3.3694 | 1.9166 |
| | MI_dir_L0.01_mean | 4.3404 | 6.0520 | 0.5129 | 0.6306 | 7.5866 | 5.5430 | 8.3124 | 5.5957 | 4.0222 |
| | H_L0.01_weight | 4.0401 | 3.5072 | 1.3166 | 3.7264 | 6.9040 | 4.4876 | 1.4611 | 2.4323 | 2.8510 |
| | H_L1_variance | 1.6115 | 0.7424 | **0.0192** | 0.3058 | 1.4813 | 1.3596 | **0.0431** | 0.7300 | 1.2880 |
| | H_L3_weight | 1.8567 | 1.9833 | 1.3437 | 0.9822 | 2.4914 | 1.6234 | 0.9984 | 1.7197 | 1.7766 |
| | HpHp_L0.01_weight | 1.4614 | 0.8984 | 0.1598 | 1.9278 | 1.8254 | 1.7748 | 0.4778 | 0.8004 | **0.2760** |
| | HpHp_L3_std | **0.0002** | 0.1060 | **0.0091** | **0.0732** | 0.5557 | 0.8084 | 0.0939 | **0.0094** | **0.0013** |
| | HpHp_L5_weight | 0.6931 | 0.3386 | **0.0129** | 0.3800 | 0.7287 | 0.5380 | 0.1935 | 0.4266 | 0.4198 |

## APPENDIX B

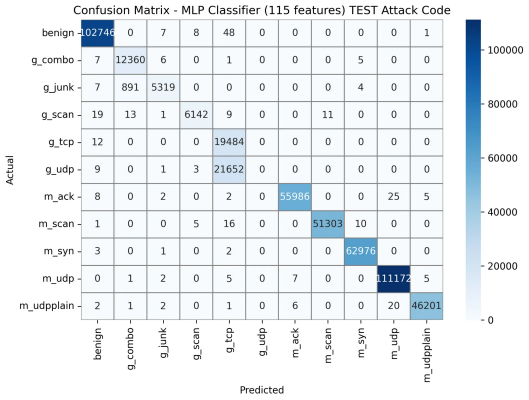


**Figure B.1 :** Confusion matrix - Attack code type (115 features).

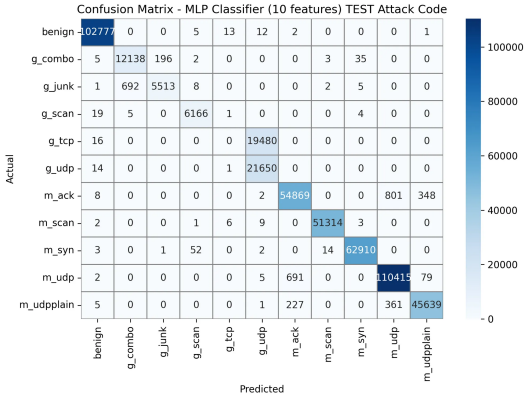


**Figure B.2 :** Confusion matrix - Attack code type (10 features).

# APPENDIX C

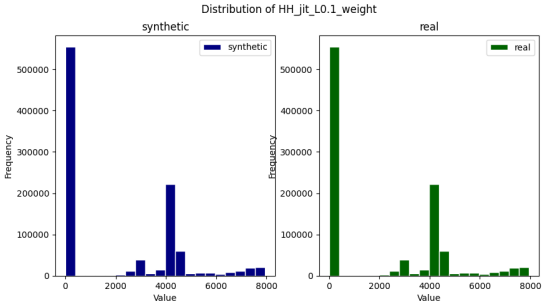


**Figure C.1 :** Distribution comparison of HH_jit_L0.1_weight.

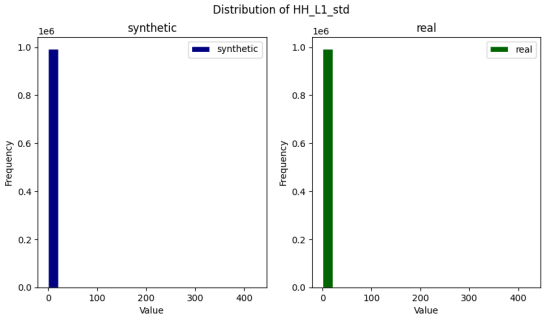


**Figure C.2 :** Distribution comparison of HH_L1_std.

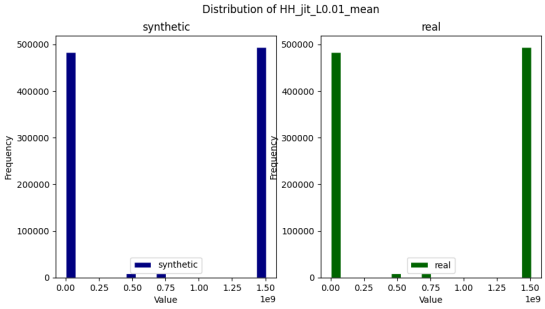


**Figure C.3 :** Distribution comparison of HH_jit_L0.01_mean.

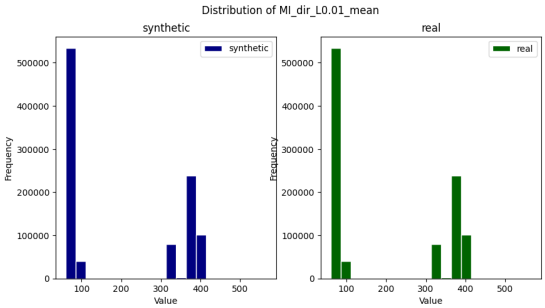


**Figure C.4 :** Distribution comparison of MI$_dir_L$0.01$_mean$.

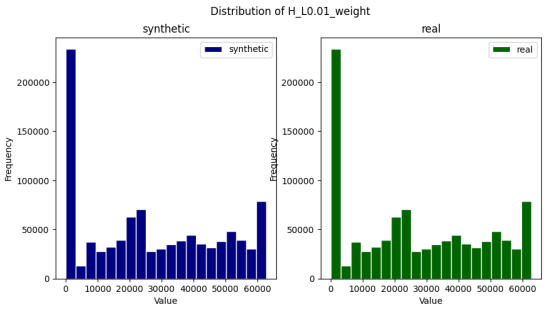


**Figure C.5 :** Distribution comparison of H_L0.01_weight.

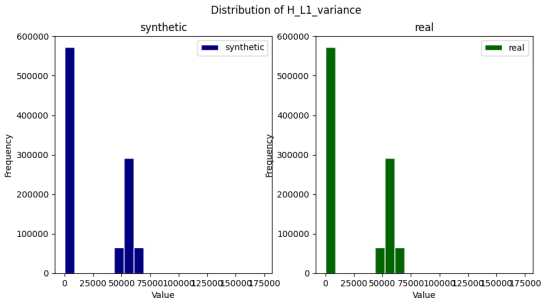


**Figure C.6 :** Distribution comparison of H_L1_variance.

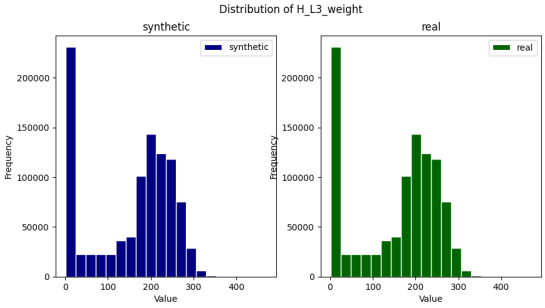


**Figure C.7 :** Distribution comparison of H_L3_weight.

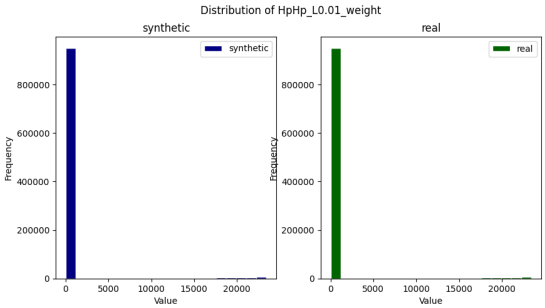


**Figure C.8 :** Distribution comparison of HpHp_L0.01_weight.

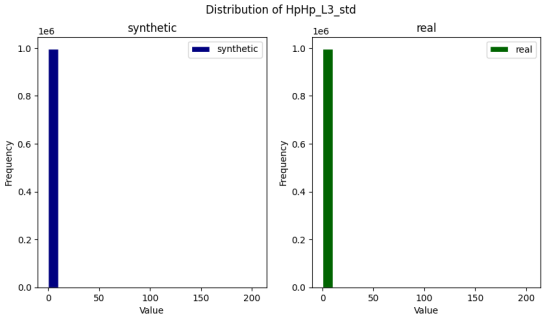


**Figure C.9 :** Distribution comparison of HpHp_L3_std.

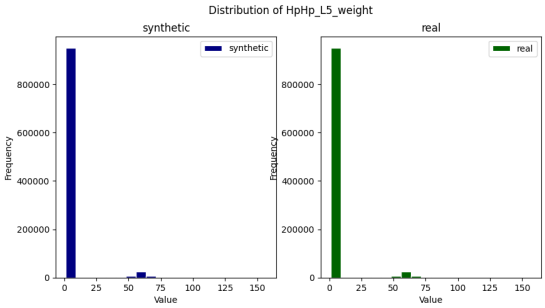


**Figure C.10 :** Distribution comparison of HpHp_L5_weight.

## CURRICULUM VITAE

**Name SURNAME:** Nilüfer USLAN

**EDUCATION:**

- **B.Sc.:** 2022, Istanbul Technical University, Faculty of Computer and Informatics Engineering, Department of Computer Engineering
- **M.Sc.:** 2025, Istanbul Technical University, Faculty of Computer and Informatics Engineering, Department of Computer Engineering

**PROFESSIONAL EXPERIENCE AND REWARDS:**

- 2023-2024 Threat Cases Operator at Cyber Struggle

**PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:**

- **Uslan N.**, Bahtiyar Ş., (2025). "Enhancing IoT Botnet Detection: A Streamlined Approach to Feature Selection", *International Graduate Research Symposium (IGRS)*, Istanbul, Türkiye, 2025, pp. 8612. (Presentation Instance)