

An Adaptive Sample-Weighting Framework for Imbalanced IoT Malware Family Classification

¹Sufyan Salim Mahmood Aldabbagh*

¹Department of Cyber Security, University of Mosul, Iraq

*e-mail: drsufyan.salim@uomosul.edu.iq

(**received:** 9 September 2026, **revised:** 23 September 2026, **accepted:** 24 September 2026)

Abstract

The extensive adoption of IoT devices raises the risk of malware attacks on IoT environments. Moreover, the increasing variety of malware families highlights the need for efficient malware family-level classification for network protection. However, the severe class imbalance problem when some major malware families represent the majority of the data while some minor families include only few instances of malware represents a challenging issue of IoT malware-family classification. Traditional machine-learning-based classifiers may demonstrate high overall accuracy but lack satisfactory performance in recognizing minority classes under imbalanced conditions. This paper introduces the approach of sample-weighting adjustment for imbalanced malware family-level classification without generating artificial instances or eliminating any existing instances. The approach is tested on the ARM subset of the CIC-YNU-IoTMal 2026 dataset, which includes 737,651 records divided between eight classes related to malware. XGBoost is used as the basic classifier model and two variations – one including all 36 features and another one with 22 features – are investigated. The accuracy of the standard 36 feature XGBoost model is 72.01%, whereas the balanced accuracy and macro-F1 score of this model are 23.37% and 20.99%, respectively, proving the insufficiency of accuracy in the presence of extreme class imbalance. On the other hand, the adaptive weight assignment technique has been able to increase the balanced accuracy to 41.43% and macro-F1 to 27.51%, which are 18.06 and 6.52 percent higher than the previous case. The 22 feature adaptive model is able to achieve similar balanced accuracy of 41.45%, proving that this smaller representation is also capable of carrying the classification capabilities of the full set of features in the context of class imbalance. Even though the adaptive method reduces the accuracy of the model, the results obtained show its capacity to improve minority class recognition without changing the size of the dataset.

Keywords: adaptive sample weighting, CIC-YNU-IoTMal 2026, class imbalance, cybersecurity, imbalanced learning, internet of things (IoT), machine learning, malware classification, malware family classification, XGBoost.

Introduction

The fast deployment of IoT devices has led to the emergence of an extremely connected computing infrastructure, where sensing, communications, and computation are gradually being integrated in normal objects and systems [1]. The IoT technology has become popular in areas such as smart homes [2], healthcare [3], industrial automation [4], transportation [5], and smart cities [6]. In addition, the existence of many heterogeneous devices and their continuous connection to the network have increased the attack surface for the cybercriminals [1]. IoT devices are continuously attacked through malicious programs, which exploit the vulnerabilities of these systems [1]. This has resulted in large-scale botnets and other forms of cyberattacks. Thus, it is vital to identify these types of malwares.

Apart from differentiating between malicious and non-malicious network traffic, malware-family classification offers even more advanced security intelligence, which includes the ability to recognize the malware family of a malicious sample. This knowledge is useful for threat analysis, incident response, tracking of malware, and proper defense mechanism deployment. Yet, multiclass IoT malware-family classification represents a significantly harder task in comparison with malware detection in general. Various malware families have similar network behaviors, whereas traffic produced by the same malware family may differ depending on the protocol used, types of attacks, and execution environment. Most importantly, the dataset containing real-world malware samples is

<http://sistemasi.ftik.unisi.ac.id>

usually highly imbalanced, in which only a few dominant malware families make up the majority of data, whereas some emerging or rarely seen malware families have only a few samples in total.

Such an imbalance results in a major issue for traditional machine learning algorithms. In the case when the majority classes predominate in the learning process, the classifier may demonstrate high accuracy while failing to classify minority classes of malware effectively. This problem is even more serious from the perspective of cybersecurity because, in this field, minority classes of malware may be considered new threats that have to be detected accurately. Thus, accuracy is not enough as the only metric for assessing multiclass malware classification with severe class imbalance.

One of the common ways to address class imbalance is to manipulate the data set during training by using methods like oversampling, undersampling, or artificial sampling generation. While this could enhance the presence of minority classes in the training set, it would change the original data distribution by including duplicate or artificial observations which do not represent actual network behavior. This becomes an even more pressing concern in the case of cybersecurity data sets, since the underlying traffic could be very complicated and hard to reproduce artificially. Another approach is cost-sensitive learning, where different observations are given different weights within the original data set. The advantage of sample weighting in this scenario is that it lets the learning algorithm concentrate on under-represented classes without actually creating new instances of observation or removing any majority class instances.

The recent advances in machine learning have shown that gradient boosting models like XGBoost can be successfully applied to structured cybersecurity data due to their capacity to capture non-linear relations between heterogeneous features of network traffic. However, the application of the standard XGBoost classifier to an extremely imbalanced dataset of malware families may lead to a significant class imbalance bias. Thus, the task is not only to achieve high classification performance but to come up with a learning approach that will increase the recognition of minority families.

The second methodology issue to be considered is the problem of data leakage in the presence of highly correlated observations. Observations within network security datasets can come from the same underlying malware sample. A random partitioning of records of the same underlying malware into training and test sets can result in an inflated generalization capability of the machine learning algorithm. A group-aware approach is therefore essential if there exists some kind of grouping feature for the observations. In the current research, the hash corresponding to the malware observation is utilized as the grouping feature and the dataset is partitioned in a way that none of the hashes will appear both in training and test sets.

In the current study, the Adaptive Sample-Weighting Framework for Imbalanced IoT Malware Family Classification is introduced and assessed based on the ARM subset of the CIC-YNU-IoTMal 2026 dataset [7]. The chosen subset includes 737,651 instances, 36 numerical features representing potential network traffic characteristics, 5,599 different hashes, and eight malware family classes. The dataset demonstrates an extremely imbalanced data structure since the largest class is represented by Mirai accounting for 68.8% of the instances while Agent accounts for just 26 instances (0.004% of the dataset). In this way, the current dataset provides a good opportunity for exploring imbalance-aware learning. The introduced framework preserves the original training samples and changes their contribution to the XGBoost learning process through adaptive sample weights. Hence, it does not require generating new instances or deleting original instances like the resampling approach.

This research also attempts to determine whether such a large feature space can be condensed while having a relatively low impact on the overall classification effectiveness. It was found through feature importance analysis that the TCP traffic was especially important, with the TCP feature contributing about 66.7% to the overall XGBoost feature importance in the first classifier. Additionally, according to the cumulative importance analysis, 22 features were responsible for about 95% of the cumulative importance. Therefore, both the full 36-feature space and the condensed feature spaces are considered.

The experiment demonstrates an interesting trade-off between majority class accuracy and minority class recognition. The regular 36-feature model had an accuracy of 72.01%, whereas the adaptive 36-feature model had an accuracy of 50.01%. The adaptive technique boosted balanced accuracy from 0.2337 to 0.4143 and macro-F1 from 0.2099 to 0.2751, which shows that the model became more balanced in terms of minority classes recognition. It means that the optimization by accuracy may overlook poor minority class recognition, and adaptive weighting can change the

<http://sistemasi.ftik.unisi.ac.id>

learning process by making it more equitable for multiclass classification. Therefore, the goal of the proposed algorithm is not to optimize accuracy at the cost of minority classes, but rather to study the alternative learning approach while keeping all observations as they were.

Summary of contributions:

1. An adaptive sample-weighting method is proposed for heavily imbalanced IoT malware family classification using XGBoost without creation of synthetic samples or removal of real samples.
2. The hash-disjoint approach is used to avoid group-level data leakage, meaning that observations with the same hash will not be present both in the training set and in the test set.
3. Feature-importance and feature-reduction analyses are performed on 36 network traffic features, as well as 22, 13, 6, and 3 reduced feature subsets to see whether there is an opportunity to create a more concise classification model.
4. Multiclass classification is evaluated under extremely heavy class imbalance using five metrics: accuracy, balanced accuracy, macro-F1, weighted-F1, and MCC.
5. The balance between traditional learning based on accuracy and adaptive learning taking into account the minority class is studied, showing that adaptive sample weighting significantly boosts balanced accuracy and macro-F1 scores.

Generally, this study provides a clear starting point for imbalance-aware classification of IoT malware family on the ARM sub-class of CIC-YNU-IoTMal 2026. While this study does not try to tackle all the problems that are associated with the detection of IoT malware, it focuses specifically on the problem of multiclass imbalance and whether adaptive sampling weight could improve detection accuracy of minority families without losing the essence of the data set.

II. Related Work

II.1 IoT Malware Detection and Classification

Great strides have been made in the last few years in detecting IoT malware using machine learning/deep learning approaches. Wang et al. (2022) used a Bi-GRU-CNN approach employing ELF binary byte sequences to detect and classify malware with high precision regardless of CPU architecture with little feature engineering [8]. Also, Wang et al. (2022) proposed a Multiscale Adaptive Attention Module (MSAAM) combined with CliqueNet for detecting malware from images with 99.8% accuracy and 99.2% family classification accuracy, and resistant to both class imbalance and obfuscation attacks [9].

II.2 Machine Learning for IoT Malware-Family Classification

New approaches based on networks and cross-architecture techniques have been proposed. Jiang et al. (2024) designed a novel Adaptive Multi-scale Dilated ResNeXt with GRU (AMDR GRU), whose classification precision for IoT malware is equal to 99.59%, implementing swarm optimization methods to fine-tune the model parameters and avoiding hand-crafted features [10]. Moreover, Alsenwi et al. (2024) suggested a cloud-assisted boosting ensemble named HierarchicalCloudDNN, which splits the learning process across embedded, edge, and cloud layers, offering an efficient approach to detecting IIoT malware (accuracy of 98.90%) [11].

II.3 Handling Class Imbalance and Performance Evaluation in IoT Security

Class imbalance continues to be an important problem. MSAAM method proposed by Wang et al. (2022) is especially designed to solve the class imbalance problem with the application of adaptive attention mechanism [9]. Li et al. (2024) investigated one-class classification models such as Isolation Forest and deep autoencoders to detect anomalies based on IoT malware using NetFlow traffic data and TF-IDF representation, emphasizing the significance of unsupervised learning for extremely imbalanced real-world IoT traffic data [12].

II.4 Approaches of Lightweight and Online Malware Detection

Efficiency and timeliness are the key factors for IoT applications. Deep Image method by Choudhary et al. (2024) converts dynamic malware behaviors into sparse binary images for online detection through deep learning [13].

II.5 Systematic Review and Open Issues

An extensive review by Aldwairi et al. (2024) reviewed deep learning approaches in IoT malware detection and forensics, noting the problems of the lack of adequate dataset, real-time solution and class imbalance [14].

III. Proposed Adaptive Sample-Weighting Framework

In this section, we present the proposed adaptive sample-weighting methodology for classifying imbalanced malware family classification problems in IoT. This methodology is intended to enhance the identification of minority malware families without synthesizing new samples or eliminating any from the existing training data. In this work, we propose the integration of adaptive sample weighting with XGBoost.

III.1 Framework Overview

The proposed framework is composed of four phases:

1. Feature preparation: The initial network-traffic features are prepared and converted to the numerical feature matrix which is employed in the classification process.
2. Group-based dataset split: The samples are split into the training and test datasets based on the malware analysis hash group so that no samples belonging to the same hash group can appear in both datasets.
3. Adaptive sample weighting: Weight is generated for each sample from the training set by taking into account how often the given sample belongs to the particular malware family. The algorithm generates more weight for less frequent malware families, but it limits the maximal possible weight.
4. Training the weighted XGBoost model: Original training samples and their weights are kept and provided to XGBoost.

The general flowchart is depicted in Figure 1:

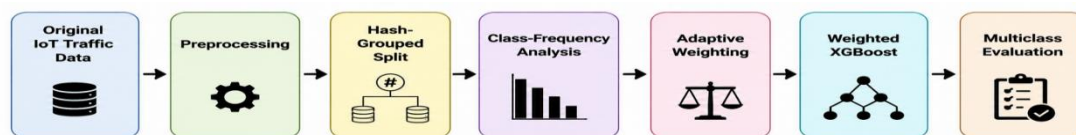


Figure 1 The overall workflow of the proposed study

Unlike resampling-based approaches, the proposed framework does not create synthetic samples and does not discard majority-class observations.

III.2 Representation of Dataset

For the experiments, the ARM part of the dataset consists of 737,651 samples and 36 numerical features after ignoring the ID and target related categorical variables. Details of the features are given in Table 1.

Table 1 Details of the features used in ARM part of CIC-YNU-IoTMal 2026 dataset

No.	Feature	Type	Feature group	Description
1	Header_Length	Numerical	Packet/Header	Length-related information derived from packet headers.
2	Protocol Type	Numerical	Protocol	Numerical representation of the network protocol used by the traffic.
3	Time_To_Live	Numerical	Network	Time-to-live (TTL) value associated with network packets.
4	Rate	Numerical	Traffic	Traffic transmission rate associated with the observed communication.
5	fin_flag_number	Numerical	TCP Flags	Number/indicator of TCP FIN flags.
6	syn_flag_number	Numerical	TCP Flags	Number/indicator of TCP SYN flags.
7	rst_flag_number	Numerical	TCP Flags	Number/indicator of TCP RST flags.

<http://sistemasi.ftik.unisi.ac.id>

8	psh_flag_number	Numeric	TCP Flags	Number/indicator of TCP PSH flags.
9	ack_flag_number	Numeric	TCP Flags	Number/indicator of TCP ACK flags.
10	ece_flag_number	Numeric	TCP Flags	Number/indicator of TCP ECE flags.
11	cwr_flag_number	Numeric	TCP Flags	Number/indicator of TCP CWR flags.
12	ack_count	Numeric	TCP Statistics	Count of ACK-related observations.
13	syn_count	Numeric	TCP Statistics	Count of SYN-related observations.
14	fin_count	Numeric	TCP Statistics	Count of FIN-related observations.
15	rst_count	Numeric	TCP Statistics	Count of RST-related observations.
16	HTTP	Numeric	Protocol/Application	Indicator or count associated with HTTP traffic.
17	HTTPS	Numeric	Protocol/Application	Indicator or count associated with HTTPS traffic.
18	DNS	Numeric	Protocol/Application	Indicator or count associated with DNS traffic.
19	Telnet	Numeric	Protocol/Application	Indicator or count associated with Telnet traffic.
20	TCP	Numeric	Transport Protocol	Indicator or count associated with TCP traffic.
21	UDP	Numeric	Transport Protocol	Indicator or count associated with UDP traffic.
22	DHCP	Numeric	Protocol/Application	Indicator or count associated with DHCP traffic.
23	ARP	Numeric	Network/Link	Indicator or count associated with ARP traffic.
24	ICMP	Numeric	Network	Indicator or count associated with ICMP traffic.
25	IGMP	Numeric	Network	Indicator or count associated with IGMP traffic.
26	IPv	Numeric	Network	Indicator associated with IP-version traffic.
27	LLC	Numeric	Link Layer	Indicator/count associated with Logical Link Control traffic.
28	Tot sum	Numeric	Statistical	Total/sum-based traffic statistic.
29	Min	Numeric	Statistical	Minimum observed value within the corresponding traffic/packet statistics.
30	Max	Numeric	Statistical	Maximum observed value within the corresponding traffic/packet statistics.
31	AVG	Numeric	Statistical	Average value of the corresponding traffic/packet measurements.
32	Std	Numeric	Statistical	Standard deviation of the corresponding traffic/packet measurements.
33	Tot size	Numeric	Traffic/Package	Total traffic or packet-size measurement.
34	IAT	Numeric	Temporal	Inter-arrival-time-related traffic statistic.

		c		
35	Number	Numeri c	Traffic Statistics	Number of observations/packets contributing to the corresponding record.
36	Variance	Numeri c	Statistical	Variance of the corresponding traffic/packet measurements.

The target variable is comprised of eight malware family classes, namely Agent, Benign, DarkNexus, Gafgyt, Generic, Mirai, Tsunami, and Unknown. There is an imbalance among the various classes, with Mirai being the major class, while Agent and Tsunami form the two very minor classes. This makes the dataset a suitable platform to test the efficacy of adaptive weighting.

III.3 Hash Aware Data Partitioning

One key problem revealed through dataset analysis is the possibility for having more than one observation that is mapped to the same malware-analysis hash. Randomly splitting the data into training and testing sets may lead to including observations with the same hash in both the training and testing datasets.

This may create a case of information leakage, where the classifier can find itself dealing with very similar observations from the same malware-instance. To mitigate the above problem, the experiment protocol has adopted hash group-level data partitioning approach. In this experiment, the generated partitions consist of:

- 4,496 hash groups for training
- 1,103 hash groups for testing
- 0 overlapped hash groups

The training partition consists of 590,120 observations, whereas the testing partition consists of 147,531 observations. This partition is considered to be more stringent compared to a regular random row split.

III.4 Class-imbalance-aware weighting

Let NN denote the total number of training samples, KK the number of malware classes, and ncn_c the number of training samples belonging to class cc . The number of samples in class cc is defined as:

$$n_c = \sum_{i=1}^N \mathbb{I}(y_i = c) \quad (1)$$

Where y_i denotes the class label of the i -th training sample and $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 when the specified condition is satisfied and 0 otherwise.

The conventional balanced class weight is calculated as:

$$w_c^{\text{base}} = \frac{N}{Kn_c} \quad (2)$$

where w_c^{base} denotes the balanced weight assigned to class c . This formulation assigns larger weights to underrepresented classes and smaller weights to majority classes.

In the current training set, class distribution is shown in Table 2.

Table 2 the distribution of the classes

Class	Training samples
Agent	21
Benign	41,432
DarkNexus	35,325
Gafgyt	8,360
Generic	7,240
Mirai	406,217
Tsunami	629
Unknown	90,896

These conventional balanced weights are highly variable. The very uncommon categories of Agent and Tsunami actually have highly inflated weights. Extremely high weights may lead to instability in the optimization process and may bias the classifier towards rare categories.

III.5 Adaptive Class Weight Transformation

To control the magnitude of the class-imbalance compensation, an adaptation factor α is introduced. The adaptive class weight is calculated as:

$$w_c^{\text{adapt}} = (w_c^{\text{base}})^{\alpha} = \left(\frac{N}{Kn_c}\right)^{\alpha}, \quad 0 < \alpha \leq 1 \quad (3)$$

where w_c^{adapt} denotes the adaptively transformed weight for class c , and α is the scaling parameter controlling the strength of the class-imbalance correction. When $\alpha=1$, the original balanced weights are preserved, whereas values of $0<\alpha<1$ reduce the difference between majority- and minority-class weights.

In particular, the conducted experiment employed $\alpha=0.75$.

This transformation lowers the extremely high weights obtained from inverse frequency calculation and yet still favors minority classes.

III.6 Maximum-Weight Constraint

Even though the transformation significantly lowers the value of extremely high weights, the extremely rare classes can have very high values. To prevent extremely rare classes from exerting a disproportionately large influence on the optimization process, a maximum-weight constraint is applied. The final adaptive class weight is therefore defined as:

$$w_c = \min \left\{ \left(\frac{N}{Kn_c}\right)^{\alpha}, w_{\max} \right\} \quad (4)$$

where w_c denotes the final adaptive weight assigned to class c , and $w_{\max}$ denotes the maximum allowable class weight. In this study, $\alpha=0.75$ and $w_{\max}=10$ were used.

The class weights obtained are provided in Table 3 below:

Table 3 Class weights

Class	Final weight
Agent	10.0000
Benign	1.5413
DarkNexus	1.7371
Gafgyt	5.1196
Generic	5.7027
Mirai	0.2782
Tsunami	10.0000
Unknown	0.8550

Such a formula is essential in the sense that it helps avoid an overly high weighting of the very rare Agent and Tsunami botnets without ignoring their significant impact relative to the dominant Mirai botnet.

III.7 Transformation of Class Weights to Sample Weights

XGBoost admits a weight for each training observation. Hence, the adaptive weight for the particular class is assigned to each observation based on its target class.

For observation i ,

$$w_i = w_{y_i}, \quad i = 1, \dots, N_{\text{train}} \quad (5)$$

where w_i denotes the weight assigned to the i -th training sample and w_{y_i} denotes the final adaptive weight associated with its class label y_i .

Thus, all observations that belong to the same malware family have the same starting class-related weight.

Hence, the weighted training set is:

$$\mathcal{D}_w = \{(\mathbf{x}_i, y_i, w_i)\}_{i=1}^{N_{\text{train}}} \quad (6)$$

where $\mathbf{x}_i$ is the feature vector of the i -th training sample, y_i is its corresponding malware-family label,

w_i is its assigned adaptive sample weight, and N_{train} is the total number of training samples.

This point is important for understanding the proposed approach. This approach does not:

- Create artificial observations;

- Repeat minority observations;
- Remove majority observations; or
- Change the initial feature vectors.

Instead, it changes the importance of each observation in the training objective of the model.

III.8 Integration with XGBoost

Adaptive weights are provided as arguments to XGBoost via the `sample_weight` parameter.

The adaptive sample weights are incorporated into the XGBoost training objective through the `sample_weight` parameter. The weighted optimization objective can be expressed as:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N_{\text{train}}} w_i \ell(y_i, f_{\theta}(\mathbf{x}_i)) + \Omega(\theta) \quad (7)$$

where $\mathcal{L}(\theta)$ denotes the overall training objective, w_i is the adaptive weight assigned to training sample i , $\ell(\cdot)$ denotes the multiclass classification loss, $f_{\theta}(\mathbf{x}_i)$ represents the prediction produced by the XGBoost model parameterized by θ , and $\Omega(\theta)$ denotes the regularization term controlling model complexity.

Consequently, samples belonging to underrepresented malware families can contribute more strongly to the optimization objective than samples belonging to the dominant class.

The key aspect of the described approach is that the initial distribution of samples remains untouched, but the optimization target is modified.

III.9 Experiments on Feature Reduction

Apart from the whole 36 features configuration, experiments on using the proposed adaptive weighting method on the feature-reduced data are performed.

According to feature importance from the usual XGBoost model, a rather small amount of network features had substantial impact on model predictions. These features included TCP, Telnet, syn_flag_number, HTTP, syn_count, IAT, Min, Header_Length, ack_count, and DNS.

On the basis of cumulative feature importance, the following four feature configurations were used for experiments:

- 36 features: complete feature candidate set;
- 22 features: features covering 95% of cumulative importance;
- 13 features: 90% of cumulative importance;
- 6 features: 80% of cumulative importance;
- 3 features: very compact.

The reduced feature configurations were evaluated to examine the effect of feature dimensionality on classification performance and to determine whether the adaptive sample-weighting framework could maintain its class-balancing behavior when fewer input features were used. In this study, the 36-feature configuration with adaptive sample weighting was considered the primary model, while the reduced feature configurations were used to assess the relationship between feature dimensionality and classification performance. It is thus important to study whether adaptive sample weight technique will be able to preserve minority class performance in case of lower feature dimensionality.

For this paper, however, the model using 36 features and adaptive sample weight is considered the main model, while the experiment using 22 features is used for preliminary feature efficiency analysis.

III.10 Experimental Baselines

The method is compared with standard XGBoost trained with hash-aware training/testing partition. The findings clearly illustrate the key trade-off: while adaptive weights significantly boost balanced accuracy and macro-F1 score, leading to better recognition of classes, they decrease the values of accuracy and weighted-F1 because the model compromises on recognizing the majority classes in favor of minority classes.

This trade-off is vital to the research problem and will be further analyzed in the next sections.

IV. Experimental Setting

This section presents the data set, preprocessing process, experimental data splitting approach, feature set, classifier parameters, adaptive weighting configuration, and evaluation measures for evaluating the proposed framework.

IV.1 Data Set

Experiments have been done using the ARM subset of the CIC-YNU-IoTMal 2026 data set. This subset includes the observations of network traffic made as a result of IoT malware-related activities on ARM-based devices. It is in the form of Parquet files and includes flow level statistics along with malware family labels and a corresponding hash identifier.

The ARM pcap.parquet data set consists of 737,651 observations and 42 features. Out of these features, 36 are numerical features related to network traffic that are candidates for predictors, while Hash, MalwareFamily, and Arch represent identifiers, targets, and architecture, respectively.

The target variable, MalwareFamily, has eight categories:

- Agent
- Benign
- DarkNexus
- Gafgyt
- Generic
- Mirai
- Tsunami
- Unknown

The data set is extremely unbalanced where Mirai is the majority class having 507,772 observations, while Agent has only 26 observations. The whole distribution is shown in Table 4.

Table 4 Distribution of malware families in the ARM data set

Malware Family	Number of Records	Percentage
Mirai	507,772	68.836%
Unknown	113,620	15.403%
Benign	51,791	7.021%
DarkNexus	44,156	5.986%
Gafgyt	10,450	1.417%
Generic	9,050	1.227%
Tsunami	786	0.107%
Agent	26	0.004%
Total	737,651	100%

This distribution clearly shows the extent of the imbalance problem tackled by this research paper. Specifically, there is a big gap between the majority class (Mirai) and minority class (Agent). Thus, accuracy cannot be the only measure of evaluation of the model's performance.

IV.2 Data Preprocessing

Preprocessing was intended to keep the original data and remove the attributes which are not supposed to serve as the predictors.

Arch was removed because all 737,651 observations belong to the ARM architecture, meaning that the attribute has just one value. Also, Hash attribute was omitted since it is used as a key to identify malware samples in the analysis teams and does not contain any information about the network behavior.

MalwareFamily was kept as the target variable.

Three more network attributes—SMTP, IRC, and SSH—were found to have just one value and were removed as constant features.

In data quality assessment, 563 instances were found to have positive infinity in the Rate attribute. At the same time, these instances had missing values for both Std and Variance attributes. Instances with such issues were detected at the preprocessing step and processed prior to model training following the described preprocessing pipeline.

Additionally, there were 3,036 fully duplicated rows in the dataset. It is important to mention that the suggested framework does not utilize random observation deletion as the class balancing technique.

IV.3 Hash-Based Group Partitioning

An important aspect of the experimental framework is the adoption of hash-aware partitioning.

In total, there are 5,599 distinct hashes. Each hash relates to a single malware family because none of the hashes are associated with more than one family of malware.

Nevertheless, the number of records linked to each hash differs widely. The median number of records for each hash is about 27, and the largest number is 41,171 records. Thus, conducting a traditional random split by rows may lead to the inclusion of the records from the same hash into both datasets.

To eliminate this potential data leakage, the dataset was split by the hash groups. This partition can be seen in Table 5:

Table 5 The partition of training and testing sets according to hash group

Partition	Hash Groups	Records
Training	4,496	590,120
Testing	1,103	147,531

Hence, the test dataset comprises unseen hash groups instead of unseen rows within seen hash groups. This method allows for a much tougher test on the generalization of malware families.

IV.4 Feature Configurations

In the first experiment of XGBoost, all 36 potential features were used. In order to evaluate whether a reduced feature set can achieve the same performance level, feature importance analysis was performed.

As a result of the feature importance analysis, the following conclusions were obtained:

- 22 features explain about 95% of the total importance;
- 13 features explain about 90%;
- 6 features explain about 80%;
- 3 features provide an extremely compact representation.

The main experiments hence involved the following configurations, as shown in Table 6:

Table 6 The configurations of the main experiments of the study

Configuration	Number of Features	Purpose
Full	36	Complete candidate feature space
Reduced-1	22	Approximately 95% cumulative importance
Reduced-2	13	Approximately 90% cumulative importance
Reduced-3	6	Approximately 80% cumulative importance
Reduced-4	3	Highly compact representation

TCP, Telnet, syn_flag_number, HTTP, syn_count, IAT, Min, and Header_Length, were some of the most important features determined by the first XGBoost model.

In the adaptive-weighting experiments, the 36-feature set-up was chosen to be the main experimental design while the 22-feature set-up was preserved for the first feature-reduction experiment.

IV.5 Baseline Classifier

XGBoost was chosen as the baseline classifier since the dataset comprises a large tabular dataset with heterogeneous numerical network flow features.

The baseline classifier was trained using a training set without any adaptive sample weighting. The aim of this classifier is to identify how well the classifier performs when it trains directly from the imbalanced dataset.

The same training and testing sets were utilized in all the experiments to make sure that any difference in the performance results was due to the weighting technique and not the split in the data set.

There were two sets of baseline experiments conducted – 36-feature and 22-feature.

IV.6 Adaptive Sample-Weighting Configurations

The proposed approach gives the weight for each training sample that depends on the class and calculated using the adaptive weighting approach outlined in Section 3.

The base balanced weight is first calculated as follows:

$$w_c^{base} = \frac{N}{K n_c},$$

and then the adaptive transformation is used:

$$w_c^{adaptive} = (w_c^{base})^\alpha$$

Finally, the maximal weight restriction is used:

$$w_c^{adaptive} = \min((w_c^{base})^\alpha, w_{max}).$$

The following two parameters regulate the weighting process:

- α – the parameter regulating the strength of the imbalance correction.
- w_{max} – the maximum possible weight of any class.

Various configurations were tested during the experiments. The chosen configuration for the main experiment with adaptive weights was:

$\alpha=0.75$, $w_{max}=10$.

The resulting final class weights for this configuration are presented below in Table 7.

Table 7 Configuration of final class weights

Class	Final Weight
Agent	10.0000
Benign	1.5413
DarkNexus	1.7371
Gafgyt	5.1196
Generic	5.7027
Mirai	0.2782
Tsunami	10.0000
Unknown	0.8550

The above-obtained weights were then mapped to the specific samples and passed to XGBoost via the `sample_weight` argument.

IV.7 Comparative Experimental Setup

The experimental setup involves the comparative study of the traditional XGBoost algorithm against the proposed adaptive XGBoost. The key experiments have been outlined in Table 8.

Table 8 Experimental setups

Experiment	Class Weighting	Features
E1	Standard XGBoost	36
E2	Standard XGBoost	22
E3	Adaptive XGBoost	36
E4	Adaptive XGBoost	22

The standard and adaptive versions employ the same hash-based split into training and testing parts, making it possible to test the impact of the weighting technique.

Feature reduction experiment aims to see whether it is possible to preserve the behavior of the method using a reduced number of input features.

IV.8 Evaluation Metrics

As the data contains many unbalanced classes, there are five metrics that complement each other.

IV.8.1 Accuracy

Accuracy is the measure of the fraction of correct classifications:

$$\text{Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{I}(\hat{y}_i = y_i) \quad (8)$$

where N_{test} is the total number of test samples, y_i is the true class label, and $\hat{y}_i$ is the predicted class label of sample i .

Although this measure is useful for evaluation in general, it may be strongly affected by the majority class.

IV.8.2 Balanced Accuracy

This metric takes each class into account equally and is computed as the average of class-wise recall:

$$\text{Balanced Accuracy} = \frac{1}{K} \sum_{c=1}^K R_c \quad (9)$$

The recall can be calculated by:

$$R_c = \frac{TP_c}{TP_c + FN_c} \quad (10)$$

where R_c denotes the recall of class c , TP_c is the number of correctly classified samples belonging to class c , and FN_c is the number of samples belonging to class c that are incorrectly classified as another class.

It is especially important in the current research as it assesses the ability to identify minority malware classes.

IV.8.3 Macro-F1

Macro-F1 is calculated separately for each class and then averaged:

$$P_c = \frac{TP_c}{TP_c + FP_c} \quad (11)$$

Where P_c is precision for class c and FP_c denotes the number of samples incorrectly predicted as class c .

$$F1_c = \frac{2P_c R_c}{P_c + R_c} \quad (12)$$

In contrast to weighted-F1, all malware families contribute equally to this score.

$$\text{Macro} - F1 = \frac{1}{K} \sum_{c=1}^K F1_c \quad (13)$$

IV.8.4 Weighted-F1

Weighted-F1 computes the F1-score for each class depending on the number of its observations.

$$\text{Weighted} - F1 = \sum_{c=1}^K \frac{n_c^{\text{test}}}{N_{\text{test}}} F1_c \quad (14)$$

where n_c^{test} denotes the number of test samples belonging to class c , N_{test} is the total number of test samples, and $F1_c$ is the F1-score of class c .

IV.8.5 Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) was chosen to serve as one more general metric because it measures the agreement between the predicted and the actual classes and is less vulnerable to the problem of class imbalance compared to the accuracy score alone.

Let $C \in \mathbb{N}^{K \times K}$ denote the multiclass confusion matrix, where C_{ij} represents the number of samples whose true class is i and predicted class is j .

$$s = \sum_{i=1}^K \sum_{j=1}^K C_{ij}, \quad c = \sum_{k=1}^K C_{kk} \quad (15)$$

$$p_k = \sum_{i=1}^K C_{ik}, \quad t_k = \sum_{j=1}^K C_{kj} \quad (16)$$

$$\text{MCC} = \frac{cs - \sum_{k=1}^K p_k t_k}{\sqrt{(s^2 - \sum_{k=1}^K p_k^2)(s^2 - \sum_{k=1}^K t_k^2)}} \quad (17)$$

where s is the total number of test samples, c is the total number of correctly classified samples, p_k is the number of samples predicted as class k , and t_k is the number of samples whose true label is class k .

This combination of metrics helps to clearly define the difference between two goals of the study:

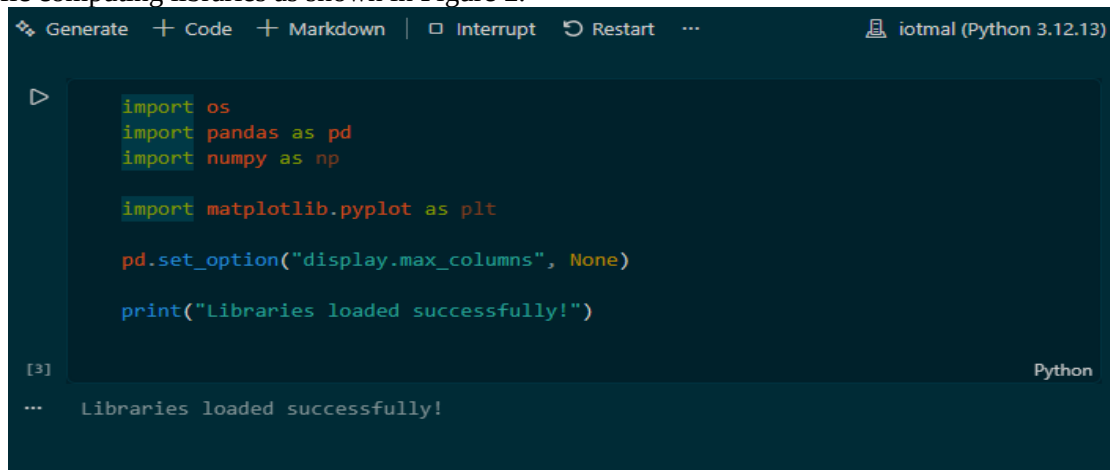
1. Predictive performance in general, characterized by accuracy and weighted-F1 scores; and

2. Identification of malware families with balanced accuracy, macro-F1, and MCC scores.

This is important for understanding the impact of adaptive weighting.

IV.9 Experiment Environment

All the experiments were conducted in Python in a special Conda environment (iotmal). The major machine learning model was built on top of XGBoost 3.3.0 with Pandas and other Python's scientific computing libraries as shown in Figure 2.



```

import os
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

pd.set_option("display.max_columns", None)

print("Libraries loaded successfully!")

```

[3] ... Libraries loaded successfully!

Figure 2 The Conda environment (iotmal) sample and the configuration of the main libraries

The data was read from the Parquet file format. Training and evaluation of models have been performed locally, and the experimental setup is carried out using Python Notebooks.

To ensure reproducibility, the paper should provide details of the versions of the other libraries used in the final experiment along with the processor, memory, operating system, and GPU information if GPU was utilized.

IV.10 Reproducibility and Experimental Fairness

For the fair comparison of the two methods, the following criteria were considered during the experimentation:

1. The same original dataset was used for all models.
2. The same hash-based train/test split was used.
3. No common hash group between training and test was allowed.
4. The same target labels were used in all experiments.
5. Standard and adaptive XGBoost models were tested using the same test observations.
6. The adaptive method changed observation weights but not added or removed training samples.
7. Several performance metrics were reported instead of focusing on accuracy only.

V. Results

In this section, the experimental results are described which were achieved through use of ARM subset from the CIC-YNU-IoTMal 2026 dataset. The aim of the evaluation is to assess the impact of the proposed sample-weighting method on the performance of the multiclass IoT malware-family classification under severe class imbalance conditions. Besides comparison of the standard and adaptive XGBoost models, the influence of feature reduction is also considered to establish if smaller number of features may achieve similar results.

V.1 Overall Classification Performance

In the first experiment, the comparison of the standard XGBoost classifier, which uses all 36 numerical traffic features, with the proposed adaptive sample-weighting approach is carried out. The results are provided in Table 9.

Table 9 Overall performance of standard and adaptive XGBoost models

Model	Features	Accuracy	Balanced Accuracy	Macro-F1	Weighted-F1	MCC
-------	----------	----------	-------------------	----------	-------------	-----

<http://sistemasi.ftik.unisi.ac.id>

Standard XGBoost	36	0.7201	0.2337	0.2099	0.6410	0.3221
Adaptive XGBoost	36	0.5001	0.4143	0.2751	0.5468	0.3130

The standard XGBoost model resulted in 72.01% overall accuracy and 64.10% weighted F1-score. Nevertheless, balanced accuracy for the standard model was very low (only 23.37%), whereas macro-F1 was even lower (20.99%). The large difference between accuracy-oriented and class-balanced metrics means that the model has been largely affected by the dominating classes.

For the adaptive XGBoost model, balanced accuracy reached 41.43% and macro-F1 became equal to 27.51%. Thus, the balanced accuracy improved by 18.06 percentage points, i.e. by about 77.28%, while macro-F1 improved by 6.52 percentage points, or by about 31.06%.

Such result is critical because the main goal of the proposed approach is not only increasing overall accuracy but improving classification of minority malware families. Hence, the introduced adaptive weighting mechanism affects the objective function of the optimization problem towards the more balanced classifier.

The price for such improvement is reduction of conventional accuracy to 50.01% from 72.01% and reduction of weighted-F1 to 54.68% from 64.10%. Such trade-off is natural when the focus is shifted towards minority classes, as it may lead to reducing the domination of the majority class among classifications.

The MCC dropped by only 0.0091 from 0.3221 to 0.3130. Hence, while the adaptive approach failed to enhance the correlation-based performance indicator, it caused a significant shift in the classification performance at the class level.

V.2 Class Level Performance

In order to discover the reason behind this discrepancy, the class-level results of the adaptive model with 36 features are provided in Table 10 below.

Table 10 Classification performance of the 36-feature adaptive XGBoost model.

Malware family	Precision	Recall	F1-score	Support
Agent	0.0000	0.0000	0.0000	5
Benign	0.5150	0.9951	0.6787	10,359
DarkNexus	0.2179	0.7088	0.3334	8,831
Gafgyt	0.0800	0.1823	0.1112	2,090
Generic	0.0597	0.6260	0.1090	1,810
Mirai	0.8581	0.4826	0.6178	101,555
Tsunami	0.0156	0.0255	0.0193	157
Unknown	0.3785	0.2944	0.3312	22,724

The adaptive model clearly exhibits a change in the behavior of class-level predictions. For example, the model was able to predict DarkNexus family with a recall of 70.88%, while in the standard model, the recall rate is just 1.74%. Similarly, the recall of Generic was improved to 62.60%, while the standard model failed to recognize this class.

The above example clearly shows that adaptive weighting is capable of improving the sensitivity of the classifier towards minority classes. However, this improvement comes with the cost of reduction in precision of several classes, which means that the model becomes more likely to predict minority class.

Another example of importance is the Mirai family, where even though it dominates the dataset, the recall rate has been reduced from 94.00% using standard XGBoost to 48.26% using adaptive weighting. However, the precision has been maintained at a high value of 85.81%. This fall in the recall rate of the majority class makes a great contribution to the reduction in accuracy.

In essence, the tradeoff between the two models has become evident.

V.3 Confusion Matrix Analysis

The normalized confusion matrix of the 36-feature Adaptive XGBoost model provides further insight into its class-specific behavior. The model achieved a recall of 99.51% for the Benign class, indicating that almost all benign samples in the test set were correctly identified. DarkNexus and

Generic also achieved substantially higher recognition rates, with recalls of 70.88% and 62.60%, respectively. In contrast, Mirai, despite being the dominant malware family, achieved a recall of 48.26%, reflecting the redistribution of classification emphasis introduced by the adaptive weighting strategy.

The recognition rates for the remaining minority classes were more limited. Gafgyt achieved a recall of 18.23%, Unknown achieved 29.44%, and Tsunami achieved only 2.55%. The Agent class was not correctly identified, resulting in a recall of 0%. This result should be interpreted cautiously because only five Agent samples were present in the test set.

Overall, the confusion-matrix analysis is consistent with the class-wise classification results and demonstrates the principal trade-off introduced by adaptive weighting: recognition of several underrepresented malware families improved, while the recall of the dominant Mirai class decreased. These findings indicate that the proposed weighting framework redistributes the learning emphasis across classes rather than maximizing performance on the majority class alone.

V.4 Effect of Feature Reduction on Classification Problem

The second experiment explored the possibility of classifying the problem based on fewer features than the 36 features used before. Based on the feature importance test, it was found that a relatively few number of features explained a lot of the cumulative XGBoost importance. TCP was the most important feature, followed by Telnet, SYN, HTTP, SYN count, IAT, and minimum packet size. Therefore, four feature reductions were analyzed, which were 22, 13, 6, and 3 features.

Table 11 Performance of standard XGBoost model on reduced features

Number of features	Accuracy	Balanced Accuracy	Macro-F1	Weighted-F1	MCC
36	0.7201	0.2337	0.2099	0.6410	0.3221
22	0.7197	0.2332	0.2094	0.6405	0.3207
13	0.7188	0.2326	0.2083	0.6395	0.3182
6	0.7122	0.2146	0.1977	0.6311	0.2849
3	0.7046	0.2402	0.1861	0.6120	0.2996

The results indicate that reducing the feature set from 36 to 22 features produced only minor changes in the performance of standard XGBoost. Accuracy decreased from 72.01% to 71.97%, Balanced Accuracy from 23.37% to 23.32%, Macro-F1 from 20.99% to 20.94%, Weighted-F1 from 64.10% to 64.05%, and MCC from 0.3221 to 0.3207. These small differences indicate that the 22-feature configuration retained performance close to that of the complete 36-feature configuration while using a reduced feature set.

This result is especially important when considering the IoT scenario because fewer traffic features may help in achieving a decrease in computational and data processing overhead.

On the other hand, very aggressive shrinking of the feature space to six or three features had led to a greater reduction in Macro-F1 and MCC. Consequently, the 22 features are a good trade-off between the two.

V.5 Adaptive Weights with 22 Features

Then the adaptive weights were used with the 22 most important features. The results are shown in Table 12.

Table 12 Comparison of standard XGBoost and adaptive one with 22 features

Model	Features	Accuracy	Balanced Accuracy	Macro-F1	Weighted-F1	MCC
Standard XGBoost	22	0.7197	0.2332	0.2094	0.6405	0.3207
Adaptive XGBoost	22	0.4916	0.4145	0.2723	0.5390	0.3117

In its turn, the adaptive 22-feature model showed a balanced accuracy of 41.45% and a macro-F1 score of 27.23% compared to 23.32% and 20.94% obtained by the regular XGBoost with all features.

Interestingly, the adaptive 22-feature model reached the same balanced accuracy level as the adaptive 36-feature model:

- Adaptive 36 features: 41.43%
- Adaptive 22 features: 41.45%

Thus, the addition of extra 14 features did not significantly improve the performance of the algorithm under adaptive learning.

However, it should be noted that the model trained on 22 features required a smaller number of input variables.

V.6 Discussion of the Main Findings

Based on the conducted experiment, three conclusions could be formulated.

Firstly, class imbalance is a significant problem that impacts the quality of the classification of different IoT malware families. The data set includes 507,772 samples of Mirai malware family and just 26 samples of Agent. Hence, a conventional classifier oriented at accuracy could demonstrate quite high accuracy, but have low classification efficiency for several minority malware families.

Secondly, sample adaptive weighting enhances class-balanced performance. The proposed approach helped to enhance balanced accuracy to 23.37% to 41.43% and macro F1 score to 20.99% to 27.51% using all the 36 features. This shows that sample reweighting can make the classifier highly responsive to underrepresented malware family classes without any change in the dataset through oversampling or undersampling.

Thirdly, feature reduction is not necessarily an impairment of the effectiveness of adaptive learning. The 22-feature adaptive learner was able to attain a balanced accuracy of 41.45% which was almost equal to that of the 36-feature adaptive model. This indicates that most of the initial features do not add much extra information once the adaptive weighting algorithm is used.

These results provide initial evidence to support the effectiveness of the proposed adaptive sample weighting approach compared to other data balancing approaches. Notably, the proposed approach preserves the original training data while only changing the weight of samples in the learning process.

The results indicate certain limitations as well. Adaptive learning technique decreases overall accuracy and weighted-F1 score significantly, whereas such rare classes as Agent and Tsunami continue to prove challenging to recognize. Such limitations cannot be considered as proof of solving the problem at hand, but can be considered as a ground for further study of adaptive weighting techniques, class-specific optimization, threshold tuning, and even advanced machine/deep learning techniques.

VI. Discussion

The results of experiments showed that the main difficulty in classification of IoT malware families was not reaching the high overall accuracy rate, but getting good discrimination in case of extremely imbalanced data sets. The results obtained from the ARM subset of CIC-YNU-IoTMal 2026 proved that there was significant difference between standard performance metrics and those based on imbalance.

VI.1 Impact of Class Imbalance

The dataset ARM shows considerable class imbalance. Mirai makes up approximately 68.8% of all the records, Unknown is approximately 15.4%, and Benign is approximately 7.0%. On the other hand, the class Agent consists of 26 samples, and Tsunami comprises 786 samples. It creates a tough environment for classification in which a regular classifier could get relatively high accuracy by predicting dominant classes and low recognition of minority classes.

This pattern is quite evident in the case of the standard XGBoost results. With all 36 features, the model got the accuracy of 72.01% and weighted F1-score of 64.10%. At the same time, the balanced accuracy and macro-F1 were 23.37% and 20.99%, correspondingly. The MCC value is 0.3221. From these numbers, it becomes evident that high accuracy does not mean balanced recognition of malware families.

It is further evidenced by the results in the classification report. While the recall for the class Mirai was 94%, several minority classes had a very low recall rate. For instance, such classes as Generic, Tsunami, and Agent were essentially unrecognized. Thus, using only accuracy as a metric is misleading.

VI.2 Effect of Adaptive Sample Weights

The proposed adaptive sample-weighting approach was meant to boost the importance of underrepresented classes during the training of a machine learning model without creating synthetic data instances or deleting existing ones. It is especially important to consider when dealing with malware databases because generated samples may not necessarily reflect network activity realistically, whereas deleted instances may contain information which would otherwise prove useful.

Adaptive approach made a huge change to the behavior of XGBoost. With all 36 features, balanced accuracy was increased from 23.37% to 41.43%, which corresponds to an absolute gain of 18.06 percentage points. Also, macro-F1 was increased from 20.99% to 27.51%, which means a gain of 6.52 percentage points. It means that the adaptive model became much more balanced than before regarding different malware families.

However, such gains in the performance were made at the expense of classical metrics. Accuracy was decreased from 72.01% to 50.01%, and weighted-F1 was decreased from 64.10% to 54.68%. Also, MCC was slightly decreased from 0.3221 to 0.3130.

Such tradeoff was inevitable. Therefore, it is essential to note that the adaptive model should not be considered a better classifier in all respects.

VI.3 Comparison Between 36 and 22 Features

There was an interesting observation from feature reduction; the performance of standard XGBoost algorithm when using 22 best features was quite comparable to that of 36-feature model:

Table 13 Comparison Between 36 and 22 Features.

Model	Features	Accuracy	Balanced Accuracy	Macro-F1	Weighted-F1	MCC
Standard XGBoost	36	0.7201	0.2337	0.2099	0.6410	0.3221
Standard XGBoost	22	0.7197	0.2332	0.2094	0.6405	0.3207
Adaptive XGBoost	36	0.5001	0.4143	0.2751	0.5468	0.3130
Adaptive XGBoost	22	0.4916	0.4145	0.2723	0.5390	0.3117

The distinction between the 36- and 22-feature standard models is insignificant. The accuracy drops only by 0.04 percentage points, whereas the decrease in macro-F1 is about 0.05 percentage points. It indicates that there is a large number of predictive features contained in the top-ranking set.

The same applies to the adaptive models. The balanced accuracy changes insignificantly, rising from 41.43% to 41.45% for reduction from 36 to 22 features. At the same time, the macro-F1 drops slightly from 27.51% to 27.23%.

It becomes especially relevant to practice: the suggested method is able to keep most of its imbalance-resistance properties with less number of input features. However, it is still impossible to say whether 22 features form an optimal solution on the basis of the current experiments.

VI.4 Interpretation of the Feature-Importance Analysis

The XGBoost feature-importance analysis demonstrated that the TCP feature is much more important than others with the importance of around 0.667. The second most important features include Telnet, syn_flag_number, HTTP, syn_count, IAT, Min, and Header_Length.

This high importance of the TCP feature is important in the sense that it means that the classification model depends mostly on the transport layer features. However, at the same time, it means that there is discriminatory information provided by other traffic features as well.

The cumulative importance analysis showed that the 22 top features comprised around 95% of the total importance. It justified the following analysis of the comparison between the complete representation (36 features) and the reduced representations (22, 13, 6, 3).

The reduction experiments have shown that the reduction to the representation of 22 features is nearly not accompanied with any performance drop in contrast with the full feature set. In addition, more aggressive reductions to 13, 6, and 3 features demonstrated decreasing performance. Thus, 22 features can be considered as a reasonable compact representation without claiming the global optimality.

VI.5 Minority-Class Recognition

The class-wise results further illustrate the effect of adaptive weighting on malware-family recognition. The 36-feature Adaptive XGBoost model achieved recall values of 99.51% for Benign, 70.88% for DarkNexus, 62.60% for Generic, 48.26% for Mirai, 29.44% for Unknown, 18.23% for Gafgyt, and 2.55% for Tsunami. The Agent class obtained a recall of 0%, although this result should be interpreted cautiously because the test partition contained only five Agent samples.

These findings show that adaptive weighting improved the model's emphasis on several underrepresented classes, particularly DarkNexus and Generic, but did not resolve the classification difficulty associated with the extremely rare Agent and Tsunami classes. Therefore, adaptive weighting mitigates, rather than completely eliminates, the effects of severe class imbalance.

VI.6 Interpretation of Results as a Whole

Taking all results into account, one can conclude about the main motive of the research and claim that class imbalance can considerably distort the performance of classifiers of malware family in IoT. The classic XGBoost demonstrates relatively high accuracy, though it shows poor performance regarding balanced accuracy and macro-F1 measures.

The adaptive sample weighting technique gives a reverse pattern. It helps improve balanced accuracy and macro-F1 measures while making the majority class less dominant. Notably, it is accomplished without data synthesis or deletion. This is a good alternative way for studying the issue of class imbalance when classifying IoT malware families by using the original dataset.

However, the results must be regarded as preliminary validation rather than a final solution of the class imbalance problem. It should be said that the decrease of accuracy and weighted-F1 implies obtaining better minority recognition but poorer majority performance. Moreover, the extremely small class of Agent still remains an issue. Therefore, further experiments with different architectures, datasets, calibration techniques, time-dependent or family-independent testing, and other ways of applying the adaptive weighting technique are required.

Nevertheless, the results obtained can be considered as an example that illustrates how adaptive sample weighting can change the class-recognition balance of an IoT malware classifier without losing initial training examples. The results obtained constitute a base for further thorough research of the issue in the course of future work.

VII. Conclusion

This research studied the problem of highly imbalanced classification of IoT malware family on the basis of ARM dataset from CIC-YNU-IoTMal 2026 collection. It was found out that the large imbalance between malware families could make standard machine learning classifiers choose majority classes, leading to high overall accuracy but low recognition of minority ones.

The standard XGBoost classifier, trained on 36 traffic features, showed 72.01% accuracy and 64.10% weighted F1-score. However, it showed just 23.37% balanced accuracy and 20.99% macro-F1. These values illustrate that the overall accuracy is not enough for proper malware-family classification in case of high class imbalance.

For tackling this problem, an adaptive sample-weighting scheme has been proposed. Instead of generating additional artificial instances or deleting some samples, this approach changes the weight of training examples based on their class distribution. The balanced accuracy of the adaptive 36-feature model has been raised to 41.43%, and macro-F1 – to 27.51%; which correspond to 18.06 and 6.52 percentage points improvement, correspondingly. These outcomes show that the use of adaptive weighting could significantly enhance the balance of classes, especially some underrepresented malware families.

In addition, feature reduction has been investigated in the course of this study. The performance of the basic 22-feature XGBoost model is very close to the full 36-feature one; namely, its accuracy, balanced accuracy, and macro-F1 have been equal to 71.97%, 23.32%, and 20.94%, correspondingly. The balanced accuracy of the adaptive 22-feature model has been nearly the same as in the case of 36-feature version, 41.45%. This shows that the selected 22 features contain the major portion of useful information for the classifier.

Nonetheless, several limitations have been discovered as well. Adaptive sample weighting has considerably lowered the overall accuracy and F1 weighted as a result of a shift in the model behavior

toward minority classes. Additionally, extremely rare families, namely Agent family, have been hard to classify due to their scarce representations. Therefore, using sample weighting does not solve the problem of information deficiency completely.

Overall, this paper has presented the initial empirical evidence of the feasibility of using adaptive sample weighting as an imbalance-aware method of IoT malware-family classification without increasing the number of training samples. This result can be used as the basis for the further exploration of imbalance-aware malware detection.

VIII. Future Work

There are several possible paths to explore further in relation to the presented research. First, the presented approach can be tested on the other architectures from the CIC-YNU-IoTMal 2026 dataset, including MIPS, MIPSEL, and X86. Second, in the future research, one can test other possible adaptations of the sample weighting and compare them with the currently presented adaptation.

Thirdly, further studies need to include comparison not only of XGBoost but also of other machine-learning and deep-learning architectures that could shed light on whether the observed effect is driven by the weighting itself or rather by the combination of the weighting and the XGBoost learning algorithm.

Fourth, there is a need for rigorous cross-dataset and cross-architecture evaluations. For example, the training of the classifier on one architecture and then the evaluation on another would provide more accurate insight into the generalization capabilities of IoT malware-family classifiers.

Finally, the extremely rare classes deserve an additional focus. Instead of ignoring these classes in evaluation, further studies need to look into techniques that can enhance their classification while taking the above observations into account. This is especially important since this kind of problem occurs in real-world scenarios when a new malware family emerges and needs to be analyzed.

This current study thus represents the first-stage study focusing on XGBoost. The observations provided herein form the methodological basis for further study including several architectures, datasets, and models.

IX. References

- [1] M. Rabbani, J. Gui, F. Nejati, Z. Zhou, A. Kaniyamattam, M. Mirani, G. Piya, I. V. Opushnyev, R. Lu, and A. A. Ghorbani, "Device Identification and Anomaly Detection in IoT Environments," *IEEE Internet Things J.*, Vol. 12, No. 10, pp. 13625–13643, 2025, DOI: 10.1109/JIOT.2024.3522863.
- [2] M. A. Abuzaraida, S. A. L. Gaud, H. A. Hneish, and Z. S. Attarbashi, "Brewing Perfection: Real-Time Monitoring of Arabic Coffee using IoT and Machine Learning," in *Selected Papers from the International Conference on Artificial Intelligence*, A. O. Albaji, Ed., *Studies in Computational Intelligence*, Vol. 1229. Cham, Switzerland: Springer, 2026, DOI: 10.1007/978-3-032-00232-7_32.
- [3] M. A. N. B. M. Tamron, Z. S. Attarbashi, M. A. Abuzaraida, N. Atitallah, S. Iftikhar, D. O. D. Handayani, and A. B. B. Basri, "IoT-based Heartbeats Monitoring System," in *Proc. 2023 IEEE 9th Int. Conf. Comput., Eng. Design (ICCED)*, Kuala Lumpur, Malaysia, 2023, pp. 1–5, DOI: 10.1109/ICCED60214.2023.10425281.
- [4] A. Alkandari, A. Alfoudery, M. A. Abuzaraida, and A. Alshehab, "Smart Automated Robot Changing Tires using Ultrasonic Sensors," *Int. J. Eng. Trends Technol.*, Vol. 71, No. 5, pp. 166–174, 2023, DOI: 10.14445/22315381/IJETT-V71I5P217.
- [5] Z. S. Attarbashi, T. A.-L. Thamodharan, M. A. Abuzaraida, S. Iftikhar, N. A. Alansari, A. B. B. Basri, and D. O. D. Handayani, "Using IoT-based Mobile Application to Build Smart Parking System," in *Proc. 2023 IEEE 9th Int. Conf. Comput., Eng. Design (ICCED)*, Kuala Lumpur, Malaysia, 2023, pp. 1–6, DOI: 10.1109/ICCED60214.2023.10425326.
- [6] M. A. Abuzaraida, N. A. H. Hilmy, N. F. M. Yaziz, and N. Alya, "IoT-Integrated Accident Detection and Automated Emergency Alert System," *International Grand Invention, Innovation and Design Expo (IGIIDEATION) 2026*, p. 124, 2026.

- [7] Canadian Institute for Cybersecurity and Yunnan University, “CIC-YNU-IoTMal 2026,” University of New Brunswick, 2026. [Online]. Available: University of New Brunswick CIC dataset website.
- [8] R. Chaganti, V. Ravi, and T. D. Pham, “Deep Learning based Cross Architecture Internet of Things Malware Detection and Classification,” *Comput. Secur.*, Vol. 120, Art. No. 102779, 2022, DOI: 10.1016/j.cose.2022.102779.
- [9] C. Wang, Z. Zhao, F. Wang, and Q. Li, “MSAAM: A Multiscale Adaptive Attention Module for IoT Malware Detection and family classification,” *Secur. Commun. Netw.*, Vol. 2022, Art. No. 2206917, 2022, DOI: 10.1155/2022/2206917.
- [10] “Classification of Malware for Security Improvement in IoT using Heuristic Aided Adaptive Multi-Scale and Dilated ResNeXt with Gated Recurrent Unit,” *Appl. Soft Comput.*, Vol. 163, Art. No. 111838, 2024, DOI: 10.1016/j.asoc.2024.111838.
- [11] M. R. B. Mosleh and S. Sharifian, “An Efficient Cloud-Integrated Distributed Deep Neural Network Framework for IoT Malware Classification,” *Future Gener. Comput. Syst.*, Vol. 157, pp. 603–617, 2024, DOI: 10.1016/j.future.2024.03.051.
- [12] T. Shi, R. A. McCann, Y. Huang, W. Wang, and J. Kong, “Malware Detection for Internet of Things using One-Class Classification,” *Sensors*, Vol. 24, No. 13, Art. No. 4122, 2024, DOI: 10.3390/s24134122.
- [13] “Deep Image: A Precious Image based Deep Learning Method for Online Malware Detection in IoT Environment,” *Internet Things*, Vol. 27, Art. No. 101300, 2024, DOI: 10.1016/j.iot.2024.101300.
- [14] S. U. Qureshi, J. He, S. Tunio, N. Zhu, A. Nazir, A. Wajahat, F. Ullah, and A. Wadud, “Systematic Review of Deep Learning Solutions for Malware Detection and Forensic Analysis in IoT,” *J. King Saud Univ. Comput. Inf. SCI.*, Vol. 36, No. 8, Art. No. 102164, 2024, DOI: 10.1016/j.jksuci.2024.102164.