IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 30, 2022, accepted June 12, 2022, date of publication June 16, 2022, date of current version June 23, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183597

Pick Quality Over Quantity: Expert Feature Selection and Data Preprocessing for 802.11 Intrusion Detection Systems

EFSTRATIOS CHATZOGLOU[1], GEORGIOS KAMBOURAKIS[2], CONSTANTINOS KOLIAS[3], AND CHRISTOS SMILIOTOPOULOS[1]

[1]Department of Information and Communication Systems Engineering, University of the Aegean, 83200 Samos, Greece
[2]European Commission, Joint Research Centre (JRC), 21027 Ispra, Italy
[3]Department of Computer Science, University of Idaho, Idaho Falls, ID 83402, USA

Corresponding author: Georgios Kambourakis (georgios.kampourakis@ec.europa.eu)

ABSTRACT Wi-Fi is arguably the most proliferated wireless technology today. Due to its massive adoption, Wi-Fi deployments always remain in the epicenter of attackers and evildoers. Surprisingly, research regarding machine learning driven intrusion detection systems (IDS) that are specifically optimized to detect Wi-Fi attacks is lagging behind. On top of that, the field is dominated by false or half-true assumptions that potentially can lead to corresponding models being overfilled to certain validation datasets, simply giving the impression or illusion of high efficiency. This work attempts to provide concrete answers to the following key questions regarding IEEE 802.11 machine learning driven IDS. First, from an expert's viewpoint and with reference to the relevant literature, what are the criteria for determining the smallest possible set of classification features, which are also common and potentially transferable to virtually any deployment types/versions of 802.11? And second, based on these features, what is the detection performance across different network versions and diverse machine learning techniques, i.e., shallow versus deep learning ones? To answer these questions, we rely on the renowned 802.11 security-oriented AWID family of datasets. In a nutshell, our experiments demonstrate that with a rather small set of 16 features and without the use of any optimization or ensemble method, shallow and deep learning classification can achieve an average F1 score of up to 99.55% and 97.55%, respectively. We argue that the suggested human expert driven feature selection leads to lightweight, deployment-agnostic detection systems, and therefore can be used as a basis for future work in this interesting and rapidly evolving field.

INDEX TERMS Intrusion detection, WiFi, 802.11, machine learning, deep learning, dataset, AWID.

I. INTRODUCTION

The ease of use and seemingly ubiquitous connectivity that Wi-Fi networks offer, have made it one of the prevalent wireless interconnection methods worldwide. From small office and home area to enterprise and public access wireless networks, the traffic generated by 802.11-enabled machines continuously increases and constitutes a significant portion of the total IP traffic. This flexibility and convenience, however, comes at the price of questionable security. Indeed, as the use of Wi-Fi is mushrooming, simultaneously 802.11-based networks are found to be susceptible to a variety of attacks at an equally rapid pace. Therefore, security of systems connected through 802.11 wireless networks becomes a subject of utmost importance.

Network perimeter defense mechanisms such as firewalls and Intrusion Detection System (IDS) play a critical role in the security of any type of communications technology and are considered as indispensable components of modern enterprise networks. Particularly, IDSs typically lie in the frontline of network security infrastructure, with the main responsibility to monitor traffic and report any perceived attack on the network. Generally, based on the underlying detection mechanism, IDSs can be categorized into misuse or anomaly detection. The former aims at distinguishing legitimate traffic from malicious based on previously identified patterns, while the latter can identify unusual deviations from a normal profile behavior. In the Wi-Fi realm, tools like Kismet [1],

The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

AirSnort [2] and ArubaOS [3] fall in the first category. In theory, such tools are advantageous in this domain, as they provide low false-positive rates. However, they never received wide adoption or the development of some of them has been abandoned. On the other hand, a critical mass of works [4]–[6] deals with the application of Machine Learning (ML) techniques for the development of misuse detection tools that are optimized for the detection of attacks in 802.11 wireless networks.

The work at hand relies on supervised methods, which need labeled network data, as those provided by the well-known AWID family of benchmark datasets. Particularly, we utilize the AWID2[1] and the newer AWID3[1] datasets with the aim to concretely answer the following key questions: (a) Based both on theory and empirical observations, are there any solid criteria to decide upon which MAC layer features are the most informative and deployment-agnostic ones for training a wireless IDS?, and (b) which is the bare minimum number of features that can yield at least fare detection rates? Overall, the main contributions of this work vis-à-vis the related literature can be outlined as follows:

- We provide a detailed reasoning behind human-driven feature selection from 802.11 frame fields. The features outlined can be used as a solid reference to the creation of robust and potentially lightweight 802.11 IDS. Furthermore, it is demonstrated that the selected features are conceivably transferable between different 802.11 datasets, possibly pertaining to diverse generations of the standard.
- We elaborate on the feature selection and data preprocessing procedures by answering the following question: Which is the bare minimum set of features that need to be considered in a 802.11 IDS implementation?
- The proposed collection of features along with the data-preprocessing scheme is assessed through an assortment of both shallow and Deep neural network (DNN) models.
- We offer a meticulous, critical review of the relevant literature, pinpointing misconceptions, half-truths, or dubious practices regarding both the feature selection and data preprocessing processes, that may lead to questionable or overfitted results.

The rest of the manuscript is structured as follows. The next section discusses the related work in this topic. Section III details the feature selection and data preprocessing process. Section IV presents the experiments, while section V elaborates on feature transferability. Section VI provides a discussion on the results, also vis-à-vis the related work. The last section concludes.

II. RELATED WORK

The current section offers a concise review of the relative work published so far considering the AWID family of datasets. We only examine major contributions in the time frame between 2017 and 2021. The focus is on feature selection and data preprocessing. Note that a more detailed, focused on particular aspects, comparison with the related work is given later in section VI. A characteristic common to most works is that they neither refer to regularization techniques nor detail their hyperparameter optimization steps.

The authors in [5] perform feature selection as a first step for tree-based classification of normal and malicious wireless traffic. Initially, they removed features that have zero values for at least 99% of the instances. Then, they proceed to Min-Max scaling. The authors conducted their experiments in AWID2 which offers independent training and testing sets. In an effort to reduce co-variance shift, they mixed samples obtained from the training and testing sets. Finally, they relied on an automated method for feature selection, namely SHapley Additive exPlanation (SHAP), to show that they can increase the accuracy (acc) particularly for identifying the impersonation class. However, in general, the features with the greatest impact as defined with SHAP are not necessarily robust and may not align with domain knowledge. Moreover, mixing training and testing sets is not valid for this benchmark dataset. Specifically, the test set in AWID2 purposefully contains unseen attacks or attacks generated with new methodologies. One of the challenges by AWID2 is to create models based on the training set alone, that generalize so that even unknown attacks such as the ones contained in the testing set can be effectively recognized as such.

The work in [7] and [8] begun with an initial step of cleaning up the dataset by eliminating features that have zero-variance. Then, the authors removed features with 50% missing values. That process alone reduced the total number of features from 154 to 36. After that, they replaced missing values for these features with the dominant value. They also replaced the MAC address with value 1 if the MAC address was a valid element. Since they did not rely on time-series analysis, they also manually eliminated all time-related features. Moreover, they discarded features that express sequence numbers. Finally, they computed the correlation between features, to identify groups of strongly positively correlated features. They chose one feature from each group and removed the rest of the features. The entire process yields a total of 18 features. We argue that missing values in AWID2 indicate that the feature is not applicable for the particular type of frame, and does not indicate some sort of measurement error. Therefore, replacing these values with the dominant one may completely alter the meaning of the field. For example, a field seen in a specific subtype of management frames (say ESSID) can never be found in frames of control type.

In [6] the authors propose an unsupervised method to detect impersonation attacks based on stacked autoencoders for feature extraction and the k-means clustering algorithm for separating malicious from benign traffic. Stacked autoencoders transform the original features into a more meaningful, low dimensional, and compact representation that captures the important information in the data. The features created

[1] https://icsdweb.aegean.gr/awid/download-dataset

after the training process are used as a new, supposedly better space for the clustering task. All 154 original features of the AWID2 dataset are first min-max scaled and balanced in an 1:1 ratio, the symbolic-valued attributes are mapped to integer values, and missing values are assigned to 0. The transformed dataset is then fed into two cascading encoders. The output of the process yields 50 features, which in turn were used for clustering $k = 2$, i.e. creating two clusters. However, most categorical features in AWID2 are not ordinal, therefore converting such features to integers and scaling them will yield decimal values. By doing so, there is high probability that the process will make the frame representation invalid and may drastically undermine the explainability of the results.

The authors in [9] first transform all features bearing hexadecimals values, e.g., MAC addresses, into the corresponding decimal representation and subsequently perform a normalization step. The next step, i.e., manual feature selection, casts off the majority of the features, leaving only 32. The value of these features gets further evaluated using the Harmony Search, AntSearch and Bee Search algorithms, and the Correlation Feature Selection measure to finally obtain alternative three sets of 5, 7, and 10 features, respectively. This work demonstrates that effective feature selection can indeed improve the performance of classification algorithms. However, the authors did not provide extensive details regarding their manual feature selection. Taking this into account, some of the features that were eliminated may be proven even useful.

Ran *et al.* [10] relied on a deep learning approach that is based on ladder neural network (NN). Specifically, the network self-learns the features necessary to detect network anomalies. In theory, deep learning can outperform shallow learning approaches in its feature learning task. The proposed network is trained to minimize both the sum of supervised and unsupervised cost functions by back-propagation at the same time. The preprocessing steps performed include (a) transformation of hex values to integers, (b) substitution of '?' with zero, (c) removal of string values, (d) removal of attributes with zero variance, (e) min-max normalization. However, preprocessing steps like the removal of string values can prove detrimental, especially for impersonation attacks.

In [11], the authors relied on a two-stage NN structure to perform classification of 802.11 traffic. The first step is based on an auto-encoder that pre-trains the network in an unsupervised way. At the second stage, a three layer NN gets trained by the output of the previous step. Preprocessing was done to replace missing values with zero. Features with duplicate information, and the features with constant values, were discarded. This resulted in a version of the dataset with 36 unique features. They also applied log encoding on the large numerical features such as source bytes, destination bytes and duration to avoid any kind of biasing. Standard scaling and one hot encoding was applied on the output labels. Unfortunately, the authors did not provide a list of the features retained. The authors compared between the training and validation sets accuracy during each epoch. Based on their results, although the validation accuracy remained constant, the training accuracy was increased after each epoch. Typically, for this type of problem, this behavior is an indication of DNN overfitting. Moreover, treating fields such as MAC addresses always has the danger of performing calculations upon these fields to produce values that do not correspond to a valid address.

Zhou *et al.* [12] proposed an IDS scheme based on the correlation of feature selection. A hybrid method comprised of the Bat Algorithm and the Correlation Feature Selection method was used to select a subset of the original features in order to improve classification performance. They tested their feature selection scheme comprising 8 features against an ensemble model. To achieve a better generalization effect, they analyzed three different datasets, including AWID2. However, as detailed in section III-A, certain features that were retained, like *wlan.ta*, can cause a bias effect, typically leading to overfitting. Moreover, data preprocessing steps, including the transformation of categorical data to numerical and then applying, say, a Min-Max scaling technique may in this context be deemed inappropriate due to the unknown correlation between the original and the transformed data. As a rule of thumb, the one-hot encoding technique should be used to convert safely categorical data into numerical.

In [13], the authors applied a mapping of the symbolic valued attributes to numeric values with normalization with the mean range method and dataset balancing. The continuous data were left as-is and the missing values were replaced with zero. The authors essentially designed an impersonation attack detector using a set of 8 features. Initially, they abstracted raw features using a deep learning technique. Next, the importance of the extracted features were measured using weighted feature selection techniques. According to the authors, the proposed scheme can be implemented in wireless network devices due to the unbalanced nature of the implementation. While, they used only the *wlan.wep.iv* feature in the Deep-Feature Extraction and Selection (D-FES) analysis, they converted it from hexadecimal to decimal values. However, such features should be handled as categorical data.

Lazar *et al.* [14] relied on Graphics Processing Units (GPU) to improve the training speed of classification training up to 65x. As a preprocessing step, the authors only scaled up the features to improve the performance of the algorithms to both training and tests sets. However, it is not clear whether they used all or relied on a subset of features when they applied their methods.

The authors in [15] proposed the first anomaly detection and classification model, capable of passively detecting Krack type of attacks that are contained in AWID3. While they presented an average accuracy of 90.15%, with the assistance of ensemble learning, i.e., XGBoost, LightGBM, and Catboost, they mostly considered custom EAPOL-based frame features. We argue that such an approach which neglects generic characteristics of the attack may lead to models that are highly adapted to the specific network, conditions,

and seen attack characteristics, as discussed in section IV. More specifically, custom features are typically very tightly connected to a specific implementation of an attack, so even tiny variations of that attack may evade detection.

Finally, works like [16]–[18] do not consider a wireless dataset, but rather one that was obtained in a wired environment (NSK-KDD dataset). We argue, that the type of traffic, and therefore the features obtained in this environment, as well as the kind of attacks that may affect such environments, are significantly different. Therefore, such works are considered out of scope of this study.

To ease the parsing of the relevant literature, Table 1 lists the features selected by the corresponding authors. A general conclusion is that the majority of the works neither refer to regularization techniques nor detail their parameters.

III. FEATURE SELECTION AND DATA PREPROCESSING

This section details the feature selection and data preprocessing procedures. The analysis takes into account the two dominant benchmark datasets for 802.11 security, namely the AWID2 and AWID3 datasets. AWID2 focuses on Wired Equivalent Privacy (WEP), Protected Access (WPA) and WPA2-Personal and contains 24 attacks and 155 features corresponding to the same number of distinct 802.11 header fields. The dataset is given in CSV format, in two different versions, i.e., the full one which spans more than 42M records and its reduced version. AWID3 is much newer, concentrating on WPA2-Enterprise and Protected Management Frames (PMF). The dataset includes 21 assaults, ranging from legacy deauthentication to more advanced and higher layer ones, such as Krack, Kr00k, amplification, malware, and botnets. It is offered in both CSV (254 features) and pcap formats (raw data, full set of all possible features).

The focus of AWID2 is on MAC layer attacks, while AWID3 contains both MAC and higher layer attacks. For this reason, only records that correspond to normal and attack classes that are common to both the datasets were retained. This means that after removing the WEP-specific attacks, i.e., the injection class, with 82,061 or ≈0.03% of the total frames, we used the reduced size versions of the datasets, namely *AWID-CLS-R-Trn* and *AWID-CLS-R-Tst* from AWID2. For AWID3, the first seven attacks (CSV files), namely, Deauth, Disas, (Re)Assoc, Kr00k, RogueAP, Krack, and Evil Twin were used, while the rest were discarded. The resulting datasets for AWID2 and AWID3 comprise a total of 2,286,766 and 15,155,345 (i.e., ≈50% of the original dataset) samples, respectively.

A. FEATURE SELECTION

We only selected PHY and MAC layer features that were selected in previous works, as stated in Table 1. Generally, feature selection applied in this work revolves around five key axes:

1) The selected features must apply to all the frame types and subtypes of 802.11. For example, such a field is the *wlan.fc.type*, because every 802.11 frame must be of one of three possible types, namely data, management, and control. On the other hand, features like *wlan.mgt.fixed.reason_code* and *wlan.mgt.fixed.beacon* are specific to management frames. The reason behind this choice stems from the information each feature may carry. Specifically, features that represent specific fields of 802.11 can decrease the generalization of a ML model. Taking the Deauthentication frame as an example, the frame must specify the reason of the requested disconnection (reason code), namely the *wlan_mgt.fixed.reason_code* feature. Therefore, if the dataset only contains samples with, say, reason code equal to 7, then an analogous rule will most likely be created in the model. For this reason, in a real-world scenario, the IDS will most likely miss attacking frames if corresponding frames include a different reason code. Overall, thirty-four features in total fall to this first feature selection axis.
2) Retained features must be bounded primarily to the theoretical foundations of the attack and not (possibly random) values of specific fields that the attacker made. Such features are neither indicative nor bounded to any attack. Examples of features that do not satisfy this criterion are the MAC address of peers, either AP or STA (fields *wlan.da* and *wlan.ta*). This is because in the captures included in the two datasets the attacker's equipment is associated with specific MAC addresses for certain attacks; For instance, for the flooding attacks in AWID3, the assailant uses the 88:66:A5:55:A2:D4 MAC address. Moreover, trivial means of spoofing the MAC address of devices exist. Therefore, it is expected that in any other situation, alternative MAC addresses will be used. Reliance upon such artifacts has the risk of creating models that are overfitted to the particular dataset. In other words, since in most datasets all the attacks stem from a specific MAC address (or a set of addresses), then algorithms may create simplistic rules like "If the MAC address is equal to xx:xx:xx:xx:xx:xx, then this frame belongs to an attack." Other prominent examples of such features are the *radiotap.datarate* and the *wlan_radio.data_rate*, both referring to the transmission speed capabilities of an STA or AP. Precisely, USB adapters, STAs in general, are not capable of achieving a similar data rate per second as compared to a dedicated AP. The data rate of a modern off-the-shelf AP can be around 60 Mbps, while that of a 802.11ac USB adapter is roughly 6 Mbps. Note that the aforesaid rates can be seen in the pcap and CSV files of AWID3, however, the data rate speed of any STA is expected to be generally much lower than that of an AP. Therefore, while the data rate static value can be useful to detect impersonation attacks, when the attacker masquerades as a legitimate AP, they may lead to biased classification models since the opponent can change its device or alter

TABLE 1. Common features of AWID2 automatically or manually selected in related bibliography. The first column contains the relevant feature name as it was extracted from Wireshark, while the last column contains the type of data preprocessing step applied.

Feature name	Description	Work
frame.time_epoch	Epoch time	[6, 12, 13]
frame.time_delta	Time delta of previous captured frame	[6, 9]
frame.time_delta_displayed	Time delta of previous displayed frame	[6]
frame.time_relative	Time since reference or first frame	[5, 6, 9, 13]
frame.len	Frame length	[7, 8, 5, 6, 9]
frame.cap_len	Frame length stored into the captured file	[5, 6]
frame.ignored	Frame is ignored flag	[9]
radiotap.length	Frame header length	[7, 8, 6]
radiotap.present.tsft	Timing Synchronization Function Timer (TSFT) present flag	[6]
radiotap.present.flags	Present flags	[6]
radiotap.present.channel	Present flag channel	[6]
radiotap.present.dbm_antsignal	Present flag dBm antenna signal	[5, 6]
radiotap.present.antenna	Present flag antenna	[6]
radiotap.present.rxflags	Present flag receive signal (RX)	[6]
radiotap.mactime	MAC timestamp	[5, 6, 9]
radiotap.datarate	Data rate value	[7, 8, 5, 6, 12, 13]
radiotap.channel.freq	Channel frequency value	[7, 8, 6, 12]
radiotap.channel.type.cck	Complementary Code Keying (CCK) flag	[7, 8, 6, 9, 12]
radiotap.channel.type.ofdm	Orthogonal Frequency Division Multiplexing (OFDM) flag	[6]
radiotap.dbm_antsignal	Antenna dbm signal value	[7, 8, 6, 9]
wlan.fc.type_subtype	Type/subtype flag	[5, 6, 9, 13]
wlan.fc.type	Type flag	[7, 8, 6, 9]
wlan.fc.subtype	Subtype flag	[7, 8, 5, 6, 9]
wlan.fc.ds	Distribution System (DS) status flag	[7, 8, 5, 6, 13]
wlan.fc.frag	More fragments flag	[7, 8, 6, 12]
wlan.fc.retry	Retry flag	[7, 8, 6, 9, 13]
wlan.fc.pwrmgt	Power Management flag	[7, 8, 6, 9]
wlan.fc.moredata	More data flag	[7, 8, 6]
wlan.fc.protected	Protected frame flag	[7, 8, 6, 9, 13]
wlan.duration	Duration value	[7, 8, 5, 6, 9, 12]
wlan.ra	Receiver address	[7, 8, 5, 6, 9]
wlan.da	Destination address	[7, 8, 5, 6, 9]
wlan.ta	Transmitter address	[7, 8, 6, 9, 12, 13]
wlan.sa	Source address	[7, 8, 6, 9]
wlan.bssid	Basic Service Set (BSS) Id	[7, 8, 6, 9]
wlan.frag	Fragment number	[7, 8, 6, 12]
wlan.seq	Sequence number	[7, 8, 5, 6, 9, 13]
wlan_mgt.fixed.auth.alg	Authentication Algorithm value	[13]
wlan_mgt.fixed.auth_seq	Authentication SEQ value	[13]
wlan_mgt.fixed.reason_code	Reason code value	[5, 6, 9]
wlan_mgt.fixed.listen_ival	Listen interval value	[9]
wlan_mgt.tim.dtim_period	Delivery Traffic Indication Message (DTIM) period flag	[9, 13]
wlan_mgt.tagged.all	Tagged parameters label	[9]
wlan_mgt.fixed.timestamp	Timestamp value	[9, 13]
wlan_mgt.fixed.beacon	Beacon interval value	[9, 13]
wlan_mgt.fixed.capabilities.preamble	Short Preamble flag	[13]
wlan_mgt.fixed.capabilities.ess	ESS capabilities flag	[13]
wlan_mgt.fixed.capabilities.short_slot_time	Short Slot Time flag	[13]
wlan_mgt.rsn.akms.type	Auth Key Management (AKM) type value	[13]
wlan.wep.iv	Initialization Vector value	[9]
wlan.wep.key	Key Index flag	[9, 13]
data.len	Data length	[5, 6, 9, 13]

the data rate anytime. On the other hand, the aforesaid two features could be used in regression ML models or to create custom features in the context of a feature engineering process.

3) Features must be applicable to both versions of the protocol, that is, must be present to both AWID2 and AWID3. For example, given that the WEP algorithm is long deprecated, the *wlan.wep.iv* and *wlan.wep.key*

features can not be seen in any of the frames provided in AWID3. What is more, the main idea is to have features that are directly applicable to as many network setups as possible.

4) Features with values that are totally or mostly invariable across every frame are disregarded. For example, in 99% of the records in AWID2, fields such as *radiotap.present.flags*, *radiotap.present.channel*, *radiotap.present.dbm_antsignal*, *radiotap.present.antenna*, and *radiotap.present.rxflags* are set to 1, while 100% of the packets in AWID3 have static values for all these fields.
5) Values for a given frame are independent to those in the previous and next frames. For instance, a feature that does not meet this criterion is the *frame.time_epoch*. These features need a data preprocessing step for converting them to a time-series; any other data preprocessing scheme is considered inappropriate and may lead to faulty classification results. No less important, such features can possibly cause a bias effect towards overfitting, since they introduce a repetitiveness. For example, an assault recorded in a dataset may transmit an attack frame every 5 normal frames (a time-series, not rate-independent event), but in a real-world scenario the aggressor can inject attack frames at will, either in a synchronous or asynchronous manner.

In addition to the above criteria, features such as the *frame.len* and *radiotap.channel.freq* can possibly identify an attack, but typically are not intentionally manipulated by the attacker. That is, while *frame.len* can indeed be altered by the attacker, it also requires changing other fields for the frame to be accepted as valid; altering a flag will not change the length of the frame. The same inference can be made for the *radiotap.channel.freq*; although the attacker can modify it, the frame will not be received by the AP, which is communicating on a different channel. In this respect and to avoid having adversarial attacks against the IDS, these features should be considered in the training set.

Based on the above-mentioned criteria, as shown in Table 2, 16 generic and common to both AWID2 and AWID3 features have been selected. In more detail, the selection of each of the selected features can be justified as follows.

1) *frame.len*: The total length of a frame can be an indication of an ongoing attack, especially when analyzing protected normal) vis-à-vis unprotected (attack) frames. For instance, with reference to AWID3 CSV file, an unprotected attack deauthentication frame may be around 86 bytes, while a legitimate (protected) one is around 102 bytes.
2) *radiotap.length*: This field refers to the size of the radiotap data. Recall that radiotap features,[2] are defined by the Wireless LAN (WLAN) driver of each wireless interface. Therefore, an attacker cannot easily alter these values. Along with the *frame.len* this field can further assist a classifier in pinpointing impersonation attacks, say, Krack. That is, looking at the AWID3 CSV file, the *radiotap.length* receives different values depending on the frame type: 56 for management and control frames, or 48, 58, or 64 for data ones.
3) *radiotap.present.tsft*: A flag-based feature, which refers to the TSFT. In an infrastructure network, APs operate as central coordinators for data distribution and power management functions. This means that the AP is responsible for maintaining the TSF time, and STAs associated with the same AP accept the AP's TSF as valid. The TSF is transmitted in the timestamp field of a beacon frame. So, if this feature is combined with others, like *frame.len* and *radiotap.dbm_antsignal*, it could assist the classifier in detecting an Impersonation or a Flooding assault.
4) *radiotap.channel.freq*: It states the radio frequency (RF) of the relevant frame, e.g., 2.4 or 5 GHz. For instance, Krack and Radio Confusion type of attacks [19] exploit different radio bands of an AP, e.g., 2472 and 5180 Hz with reference to the Krack and Deauth attack files of AWID3, respectively.
5) *radiotap.channel.type.cck*: It is mainly used with 802.11b, to indicate if the current channel uses the CCK[3] or not. The CCK defines the length of the transmitted data. As with the *radiotap.channel.freq*, the current feature can be used in combination with other features to highlight different channel attacks, say, a 802.11b network operates on the 2.4 GHz band, while a 802.11ac may operate on the 5GHz band as well. Therefore, as demonstrated in [19] (refer for instance to the so-called "Radio confusion" and "Radio confusion revisited" attacks), this feature can

TABLE 2. The 16 selected features and the data preprocessing method applied to each one. OHE stands for one-hot encoding.

Feature name	Preprocessing	Also used in
frame.len	Min-Max	[7, 8, 5, 6, 9]
radiotap.length	Min-Max	[7, 8, 6]
radiotap.dbm_antsignal	Min-Max	[7, 8, 6, 9]
wlan.duration	Min-Max	[7, 8, 5, 6, 9, 12]
radiotap.present.tsft	OHE	[6]
radiotap.channel.freq	OHE	[7, 8, 6, 12]
radiotap.channel.type.cck	OHE	[7, 8, 6, 9, 12]
radiotap.channel.type.ofdm	OHE	[6]
wlan.fc.type	OHE	[7, 8, 6, 9]
wlan.fc.subtype	OHE	[7, 8, 5, 6, 9]
wlan.fc.ds	OHE	[7, 8, 5, 6, 13]
wlan.fc.frag	OHE	[7, 8, 6, 12]
wlan.fc.retry	OHE	[7, 8, 6, 9, 13]
wlan.fc.pwrmgt	OHE	[7, 8, 6, 9]
wlan.fc.moredata	OHE	[7, 8, 6]
wlan.fc.protected	OHE	[7, 8, 6, 9, 13]

[2]https://www.radiotap.org/

[3]After Wireshark v1.12.13, the keyword "type" has changed to "flags" for the *radiotap.channel.type.cck* feature. As a result, this feature is named *radiotap.channel.flags.cck* in AWID3.

aid in identifying impersonation or flooding assaults unfolding simultaneously over different channels.

6) *radiotap.channel.type.ofdm*: It is used to state if the radio channel uses OFDM modulation.[4] The current feature defines the frame structure in 802.11a/g type of 802.11 networks, and serves a similar goal to the *radiotap.channel.flags.cck* one. As with the two previous features, the OFDM channel flag is included to assist other features into detecting channel type of attacks.
7) *radiotap.dbm_antsignal*: It denotes the radio frequency signal power of each antenna in decibels. This feature can be used as a means to observe opponents that reside in a further position than a legitimate STA [20]. Simply put, significant differences in signal power along with other features, say, the *frame.len* may provide indication regarding a spoofed packet.
8) *wlan.fc.type* Naturally, the type of each frame, e.g., Management can assist into discerning malicious frames vis-à-vis normal traffic.
9) *wlan.fc.subtype*: It represents the subtype of the frame, say, a beacon frame. Along with the *wlan.fc.type*, the current feature can designate different subtypes of frames, say, a beacon frame (1000) and a QoS data frame (1000); however the first is a management (01), while the second a data (10) frame.
10) *wlan.fc.ds*: The direction of a frame can signify an impersonation attack, such as an Evil Twin. For example, an STA will never transmit a frame that it would usually be sent by an AP.
11) *wlan.fc.frag*: A flag that designates if the previous transmitted data frame is being sent in fragments. This flag may be of aid in detecting impersonation type of attacks. For instance, with reference to the AWID3 dataset, in Evil-twin attacks this field is always 0, while in some Krack frames it receives the value of 1.
12) *wlan.fc.retry*: The retry flag is set in transmissions. In this respect, it can pinpoint flooding attacks that typically transmit a spray of identical frames. For example, Deauthentication and Disassociation attacks can be identified through this flag.
13) *wlan.fc.pwrmgt*: The power management flag designates if a STA has entered the power management state or not (idle state). When a STA is in idle state, it will not transmit or receive data frames. As a result, this field can be used to identify an attacker who impersonates an idle STA. In some rare cases, a STA which requests data frames and has the current flag disabled, will not transmit Deauthentication or Disassociation frames, which in turn may indicate a flooding attack. It is important to say that in both the datasets this field is set or not for both normal and attack traffic, hence an adversarial attack on it is cumbersome.
14) *wlan.fc.moredata*: By setting the current flag, an AP can notify an STA that at least one additional Bufferable Unit (BU), i.e., a data or management frame, is buffered for it at the AP. A rogue AP will usually transmit unprotected data frames, which will have in some cases this flag enabled. If combined with the *wlan.fc.protected* flag, this behavior can pinpoint an impersonation attack. As with the previous field (13), in both the datasets, the current one is set or not in both normal or attack frames.
15) *wlan.fc.protected*: The protected flag is set to inform about whether a frame is unprotected or it has been cryptographically protected. Note that with the advent of the IEEE 802.11w amendment (included in 802.11-2016 and 802.11-2020 standards and mandatory for the WPA3 certification), sensitive management frames are cryptographically protected. The same applies to the beacon protection scheme introduced in IEEE 802.11-2020 standard. This can be of significant importance when identifying flooding or impersonation attempts.
16) *wlan.duration*: Its contents vary with frame type and subtype and in conjunction with the Quality of Service (QoS) capabilities of the sending STA. For a QoS STA, this value is the amount of airtime in microseconds the sending radio is reserving for the pending acknowledgement frame. By combining information stemming from the current and the *radiotap.dbm_antsignal* features, it may be easier to identify flooding and impersonation attacks. Namely, as with the *radiotap.dbm_antsignal*, a rational assumption is that often attackers reside in a further distance than the legitimate STAs, and as a result, attack frames may have higher *wlan.duration* values. Note that while the current field is time-based, it was included in our analysis, since the duration value does not have ordinal characteristics and is independent of previous or subsequent frames. In contrast, fields like *wlan.seq* are considered to be a time-based feature, since each value is related to the previous and the next ones.

B. DATA PREPROCESSING

The data preprocessing procedure concentrates on the encoding, normalization, and standardization techniques and is tightly connected to the theoretical background of each selected feature as given above. As shown in Table 2 each feature type went through a different conversion technique. Consequently, we employed two commonly used data encoding/scaling techniques, namely, *One-Hot Encoding (OHE)* (for features represented by discrete values), and Min-max scaling (for features with discrete numeric values). The same approach was followed for both datasets. That is, both datasets were analyzed "as is," without altering their imbalanced nature, say, through a sampling technique.

Next, for both the datasets, each CSV file was searched via the Python *Pandas* library to find any undefinable

[4] After Wireshark v1.12.13, the keyword "type" has changed to "flags" for the *radiotap.channel.type.ofdm* feature. Therefore, this feature is named *radiotap.channel.flags.ofdm* in AWID3.

values, including "Null," and "NaN" ones, or minuscule decimal values, i.e., those expressed in scientific notation, e.g., 4.854825e-1; such values produce errors in shallow and DNN analysis. Altogether, for AWID3 these values were found to be only the 0.02% or 419,504 frames of the dataset, so we removed the corresponding rows; note that for AWID2 the corresponding number of rows were much less. Three classes were defined for AWID3: *Normal*, *Flooding*, and *Impersonation*, matching the seven MAC layer attacks available in that dataset. The same classes are predefined in AWID2; the rows belonging to the Injection class, also present in AWID2, were disregarded. For instance, for AWID3, the *Flooding* category contains Deauth, Disas, (Re)Assoc, and Kr00k attacks, while the *Impersonation* includes RogueAP, Evil_Twin, and Krack.

IV. EXPERIMENTS

The current section details the methodology and presents the derived results both for shallow and DNN classifiers. As stated in section III-B, we implemented the fewer possible changes to both the datasets to ensure IDS generalization. We also relied on commonly accepted ML techniques, without resorting to any optimization or dimensionality reduction techniques. The main focus is to manually choose features that generalize well for known and unknown attacks, and across old and modern versions of the standard. A side goal is to evaluate whether expert-driven feature selection outperforms NN-based methods which perform this step automatically, under-the-hood. With reference to sections II and III, this is done to support the assertion that with a small set of attack-independent features and not optimized classification methods, the IDS can yield acceptable results. With this aim in mind, we did not resort to hyperparameter optimization methods such as the *Grid search* or *Optuna*, to possibly obtain the optimal results of each ML algorithm, but basically relied on trial and error. The reader should consider that achieving the maximum accuracy with the identified features is not the primary objective of the experiments. Our main intention was to evaluate the transferability of the selected features across different network conditions. With this in mind, we still made the best effort to conduct and report only the evaluations that achieved significantly high predictive accuracy. Specifically, in regard to the followed methodology, the next points are of essence.

- All the ML algorithms employed for evaluation purposes along with their respective setup parameters were common for both the datasets and across all the experiments for the same type of analysis. This was done to evaluate the generalization properties of the selected features.
- Because the datasets are unbalanced, we focus on the AUC and F1 metrics.
- The selection of the ML algorithms was based on two criteria. First, with an eye towards reproducibility purposes, that is, the implementation of each selected algorithm is readily available in well-known ML libraries. Second, after selecting and conducting preliminary tests on the most common ML algorithms, we only kept the ones presenting both a good combination of AUC/F1 scores and a fast training time. For instance, after these preliminary tests, Naive Bayes and AdaBoost were left out because they demonstrated a low performance, i.e., a F1 score of ≈71% and ≈82%, respectively.
- Given that AWID3 has much more samples, and it was created with the PMF always active, a logical assumption (which is further validated in section VI) is that this dataset will produce somewhat better classification results compared to those of AWID2. Recall that PMF protects against deauthentication and disassociation attacks, therefore, as detailed in §5 of [21] and in [22], an attacker would need to persistently spray with unencrypted Deauthentication and Disassociation frames in an attempt to achieve DoS or choose other avenues, say, PMF protected deauthentication frames that exploit a zero-day vulnerability as given in §6 of [19]. Based on the previous assumption, following a trial-and-error approach, we first examined the shallow classification of AWID2. This was done to find satisfactory results, i.e., an F1 score above 90%, and if possible, by using identical parameters across all ML algorithms of the same type. After an ample set of algorithms was found, we executed them using the exact same parameters against AWID3. A similar approach was followed for the DNN analysis as well.
- For shallow analysis, we made sure the training accuracy is always below or very near (±0.05%) to the testing accuracy, while for DNN we made the same check, but with reference to the loss function.
- To avoid overfitting, diverse regularization parameters and techniques were implemented.
- Because the datasets are imbalanced, we used the stratified k-fold validation (val) method with the k parameter equal to 10; this will ensure that every k-fold test set will receive the exact same number of samples from each class of the dataset. It is to be noted that for creating the 10-fold validation sets for AWID2, we merged the training and testing sets into a single dataset.
- In DNN analysis, a part of each fold (20%) or ≈411,617 samples for AWID2 and ≈2,727,962 for AWID3 were used as a validation test. This percentage of the samples was removed from the training set of each fold. Consequently, each fold in DNN analysis incorporated fewer samples in comparison to that used in swallow analysis. Recall that the validation test can aid in avoiding overfitting.

A. SHALLOW CLASSIFIERS

Shallow classification on the two AWID datasets were conducted against seven popular ML models, namely Logistic Regression, LinearSVC, Stochastic Gradient Descent Classifier (SGDClassifier), Light Gradient Boosting Machine (LightGBM), Decision Trees (DT), Random Forest (RF), and Extra Trees (ET). This was done in an effort to comparatively

present the best performers considering the above-mentioned methodology and under the 16 common features given in subsection III-A. The classifiers were implemented on an MS Windows 10 Pro AMD Ryzen 7 2700 CPU machine with 64 GB RAM. All the experiments of the shallow classification were done on the CPU; no GPU was utilized. Table 3 includes the parameters used per classification model. We relied on *sklearn* v.1.0.1 in Python v3.8.10, for all classifiers and metrics, but lightGBM. The latter algorithm was implemented with the Python library *lightgbm* v.3.3.2.

1) CONFIGURATION OF PARAMETERS

The analysis is focused on three different shallow classifier categories, namely, stochastic, linear, and tree-based ones. From the first category, we consider Logistic Regression and SGDClassifier, from the second LinearSVC, and from the third LightGBM, DT, and RF.

The Stochastic-based algorithms are dependent on the Stochastic Gradient Descent (SGD) function to train their data. In this work, we used the optimizer Stochastic Average Gradient (sag) solver to assist the Logistic Regression in predicting each class. Therefore, the multi class parameter was set to *multinomial*.

The maximum iterations' parameter was set to 1000, determining the number of iterations the solver will have to perform until convergence. Moreover, the tolerance (tol) parameter was set to 0.01; this adds a regularization effect to the training process. Additionally, the Logistic Regression implements the L2 regularization method by default. The latter method is similar to the Ridge regression, i.e., it adds a penalty equal to the square of the magnitude of coefficients.

The SGDClassifier is a variant of Logistic Regression with sag solver. This algorithm uses a linear model and the SGD properly adjusts the learning rate of the model. In our case, the modified *Huber* loss function was used. This means that the current algorithm predicted in a one-vs-all classification fashion. To avoid overtraining the model, the early stopping parameter was enabled. The validation set was assigned with the default value of 0.1, that is, the 10% of the training set was removed and used as the validation set.

With respect to the regularization effect, the current model implemented by default the *alpha*, *penalty*, and *tol* values equal to 0.0001, l2, and 1e-3, respectively. The tol value was reduced to 1e-05; this lessens the regularization effect of the current model.

Regarding LinearSVC, it was configured to use the (by default) square hinge loss function, i.e., the square of the standard SVM loss function. To improve the testing F1 score, the maximum iterations were set to 20K, and to soften the regularization effect, we increased the *C* value from 1 to 1.5. Similarly to the SGDClassifier, the current model was implemented the by default one-vs-rest classification scheme.

Tree-based algorithms are more prevalent in the public literature, with LightGBM being the newest among them. For this classifier, the default boosting parameter type, namely Gradient Boosting Decision Tree (*gbdt*) was used. LightGBM can cause an overfitting effect, especially if the samples are low (<200K). For this reason, we used diverse regularization parameters and techniques. First, we determined the level of the tree by setting the *max depth* parameter equal to 10. By having a specific length in conjunction with a proper taxonomy across all leafs, it can lead to the creation of a more generalized model. This methodology was common across all the Tree-based algorithms. Therefore, the number of leaves was configured to 20. For the same reason, the *maximum bin*, *minimum data in bin*, *minimum number of child samples*, and *minimum split gain* parameters were set to 20, 10, 30, and 0.1, respectively.

LightGBM stores the data into bins before learning them. Smaller values for the *max_bin* parameter can assist in decreasing the training time. This is due to the splitting procedure that takes places after each bin. If the LightGBM makes a split, a new leaf is created. For this reason, we increased by a small percentage the minimum split gain parameter (from 0.0 to 0.1), basically, forcing LightGBM to be more precise when creating a new leaf. The same logic was followed with the minimum number of child samples. Increasing this parameter can force a leaf to contain more data, i.e., creating a more generalized model. The remaining parameters for LightGBM include the learning rate, the number of estimators, the reg alpha, and lambda. The first is the rate in which the model will be trained to reduce the loss function; a smaller learning rate can achieve better learning capabilities. We assigned the value of 0.01 for this parameter. The value of 80 was selected for the number of estimators, which represent the number of boosted trees to fit during the training phase. *Reg alpha* and *lambda* add a regularization effect on the weights of the model, L1 and L2, respectively. Both these parameters were set to 0.01.

Similarly to LightGBM, the DT was configured with a maximum depth of 20 and a maximum number of leafs equals to 100. The default value of the minimum samples per leaf was doubled (set to 2), for forcing each leaf to collect more information. Moreover, the *ccp alpha* complexity parameter was used for adding a minimal cost-complexity pruning effect. This parameter, set to 0.001, chooses the subtree with the largest cost complexity that is also smaller than the value of *ccp alpha*.

Regarding RF, the same values per parameter as with the DT were tested, showing promising outcomes. Recall that the main goal of this work is not to achieve optimal results, but to propose a generalized methodology for supervised learning in 802.11 intrusion detection.

Lastly, ET is similar to RF, with the difference that it uses the whole dataset during the training phase. Interestingly, as observed from Table 3, the this classifier needed more estimators (trees), a deeper depth and more leafs, to perform well.

2) RESULTS

Tables 4 and 5 group the shallow classification results of the two AWID datasets; the results represent the average

TABLE 3. Parameter values per classification algorithm. A hyphen denotes that the current value is either inapplicable to the current ML algorithm or, if it is applicable, it implements the default value.

Parameters	Stochastic-based		Linear-based	Tree-based			
	Log. Regression	SGDClassifier	LinearSVC	LightGBM	DT	RF	ET
solver	sag	–	–	–	–	–	–
max_iter	1000	–	20000	–	–	–	–
tol	0.01	1e-05	–	–	–	–	–
C	–	–	1.5	–	–	–	–
loss	–	modified_huber	–	–	–	–	–
early_stopping	–	True	–	–	–	–	–
learning_rate	–	–	–	0.01	–	–	–
max_bin	–	–	–	20	–	–	–
max_depth	–	–	–	10	20	20	200
min_child_samples	–	–	–	30	–	–	–
min_data_in_bin	–	–	–	10	–	–	–
min_split_gain	–	–	–	0.1	–	–	–
n_estimators	–	–	–	80	–	–	200
num_leaves	–	–	–	20	–	–	–
reg_alpha	–	–	–	0.01	–	–	–
reg_lambda	–	–	–	0.01	–	–	–
n_jobs	–	–	–	1	–	–	–
ccp_alpha	–	–	–	–	0.001	0.001	0.0001
max_leaf_nodes	–	–	–	–	100	100	500
min_samples_leaf	–	–	–	–	2	2	2
min_samples_split	–	–	–	–	–	–	10

score calculated over all folds. Specifically, each Table contains the most relevant evaluation metrics per classifier, namely AUC, Precision, Recall, F1-Score and Accuracy, along with the total time of each model's execution in hours/minutes/seconds. Note that the Accuracy column is included just for reasons of completeness and therefore shown in gray background. The best case with regard to the AUC and F1 scores per classifier is highlighted with green, whereas the worst case with orange. Moreover, the best performer across all the classifiers in terms of average AUC is shown in green font. As an adjunct to commenting on the presented results of the classifiers, section VI present the exported, average confusion matrices per ML algorithm analysis for both the datasets.

With reference to AWID2, the best mean results were achieved with the DT algorithm. As presented in Table 4, the aforesaid classifier achieved a mean AUC of 95.16%. At the same time, LightGBM and Logistic Regression classification models succeeded the second and third-best average accuracy score, with 94.41% and 87.56% percentage each. On the other hand, RF did not manage to exceed 86.51% in all its successive executions, whereas SGDClassifier had the worst performance, namely, an average AUC of 85.99%. With regard to the total time of execution of each model, SGDClassifier was the fastest algorithm, with ≈5 min of training time, while RF revealed the maximum delay during the training of its model with ≈1 hour.

With respect to AWID3, the ET classification model scored the best mean AUC of 99.49%. LightGBM managed the second-best accuracy score, with an average value of 99.42%.

TABLE 4. AWID2 results of shallow classifiers analysis. T.E. time stands for total execution time.

Model	AUC	Prec	Recall	F1	Acc	T.E.Time
Logistic Regr.	87.56	97.94	83.36	89.57	98.55	00:10:04
SGDClassifier	85.99	89.24	81.36	84.56	97.70	00:05:18
LinearSVC	86.07	98.64	81.37	88.59	98.44	00:42:14
LightGBM	94.41	98.63	92.47	95.36	99.31	00:10:33
DT	95.16	93.71	92.93	93.31	99.11	00:08:56
RF	86.51	98.89	81.98	89.10	98.49	01:02:52
ET	94.07	98.77	92.05	95.18	99.28	02:19:07
AVG	89.96	–	–	90.81	–	–
STDEV	4.00	–	–	3.67	–	–
AVG (best 3)	94.54	–	–	94.61	–	–

On the negative side, as with AWID2, SGDClassifier had the worst mean AUC score failing to exceed an average of 95.17%, however it was the fastest with ≈27 min of training time. Regarding RF, while its model managed to reach an average AUC of 95.85%, that was attained with the worst training time of ≈5 hours.

More importantly, focusing on the three bottom-most lines of both Tables 4 and 5, it can be deduced that the average (AVG) AUC and F1 scores for the examined datasets present a difference of roughly 8% for both these metrics; the same difference is narrowed down to roughly 4.5% if considering the best three classifiers. This result suggests that the selected features are indeed applicable to both versions of the protocol. Recall that AWID3 was anticipated to produce somewhat better and more cohesive classification results compared to those of AWID2, due to reasons explained at the start of the current section and further analyzed in section VI.

TABLE 5. AWID3 results of shallow classifiers model analysis.

Model	AUC	Prec	Recall	F1	Acc	T.E.Time
Logistic Regr.	98.37	99.63	97.96	98.77	99.87	01:17:51
SGDClassifier	95.17	99.62	93.96	96.60	99.65	00:26:59
LinearSVC	98.57	99.73	98.13	98.92	99.89	01:02:07
LightGBM	99.42	99.89	99.22	99.55	99.96	00:51:34
DT	98.24	99.70	97.65	98.66	99.88	00:41:38
RF	95.85	99.76	94.16	96.80	99.76	05:38:09
ET	99.49	99.75	99.28	99.52	99.96	11:39:30
AVG	97.87	–	–	98.40	–	–
STDEV	1.57	–	–	1.12	–	–
AVG (best 3)	99.16	–	–	99.33	–	–

B. DEEP NEURAL NETWORKS

Regarding DNN analysis, using the same 16-features set given in Table 2, we created two different models, namely, Multi-layer Perceptron (MLP), and Denoising stacked Autoencoders (AE). The experiments were carried out on an MS Windows 10 Pro AMD Ryzen 7 2700 CPU machine with 64 GB RAM and a GTX 1060 6 GB GPU. We employed the *sklearn* v.1.0.1 for all classifiers and metrics, *keras* v.2.8.0, and *tensorflow* v.2.8.0-dev20211113 in Python v3.8.10. Moreover, to speed up the training process, processes were offloaded to a GPU using the CUDA v11.0 API.

1) CONFIGURATION OF PARAMETERS

Table 6 contains the parameters used for each of the DNN models. For obtaining complete control over the training phase, we implemented the mini-batch SGD optimizer, with the learning rate of 0.01 and a momentum of 0.9. A low batch size, say, 150 can result in a more generalized DNN model, since more data will be analyzed during each *epoch*. For this reason, a *batch* size of 32 and 200 were used for AWID2 and AWID3, respectively. Moreover, we exploited the well-known *Rectified Linear Unit* (ReLU) activation function, where applicable. Another common activator function for the output layer of DNN is the so-called *Softmax*. The latter was implemented to classify our results. To add a regularization effect, we relied on the *Dropout* technique.

Since Dropout randomly disables nodes of the DNN during the training phase and enables them again during the testing phase, it is logical for the validation accuracy to be higher than the training accuracy, as showed in Figures 1 to 2.

For both DNN of MLP and Denoising stacked Autoencoders, the input layer was different per dataset, i.e., 51 and 43 columns for AWID2 and AWID3, respectively. The output was the three classes defined in section III. Additional techniques, including *Model Checkpoint* and *Early Stopping*, were utilized to keep the optimal training state per DNN model. And along with other techniques, including, *Dropout* and *validation test*, we kept overfitting to the bare minimum. Regarding Model Checkpoint and Early Stopping, we checked for the minimum loss value, and if the DNN model did not improve their loss value for two consecutive epochs, it stopped the training phase and was re-trained with the last optimal epoch. This eventually means that every fold was trained for at least two more epochs. For the interested reader, an example of the usage of the *Early stopping* method can be seen in the left part of Figure 1, in which the model started after the 25th epoch to become overfit; the validation loss started to approach the training loss. As a result, the training stopped at the 31st epoch, and got retrained until the 29th one, thus possibly avoiding overfitting.

TABLE 6. Parameter values per DNN algorithm. The values of "/3" and "/2" in the MLP dropout parameter indicate the number of layers in which this parameter had the designated value. The layer values are calculated without including the input and output ones. A hyphen defines a non-applicable option for this DNN model. AE and SCC stand for sparse categorical crossentropy and autoencoders, respectively.

Parameters	MLP	AE
Activator	ReLU	ReLU
Output activator	Softmax	Softmax
Initializer	He_uniform	–
Optimizer	SGD	SGD
Momentum	0.9	0.9
Dropout	0.25/3-0.2/2	0.25
Learning rate	0.01	0.01
Loss	SCC	SCC
Batch Norm.	Yes	Yes
Hidden layers	5	7
Nodes (Per layer)	30/20/16/12/6	20/15/10/5/10/15/20
Batch size	32 for AWID2 and 200 for AWID3	

2) RESULTS

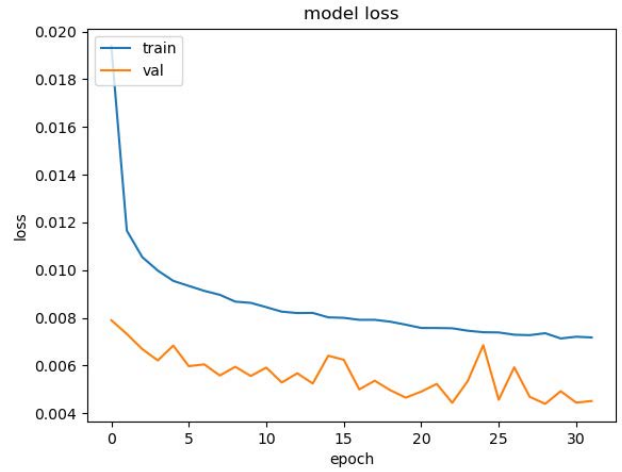
The results of the worst and best validation folds in terms of the AUC metric per examined model are presented in Table 7 and 8 for AWID2 and AWID3, respectively. Each Table contains the number of epochs needed by the relevant DNN model to be trained, along with other evaluation metrics, namely, AUC, Precision, Recall, F1, and Accuracy score. Additionally, section VI contains the results of each confusion matrix 7 and 8 Figures.

Regarding AWID2, the best average AUC score was achieved by MLP, with an average value of 81.79%. Also, this DNN model was the fastest in terms of training time, i.e., ≈20 hours. The Denoising stacked Autoencoders presented the worst performance, namely, an average AUC of 80.07%. Generally, the number of optimal epochs in each best fold was mostly around or below 10 epochs.

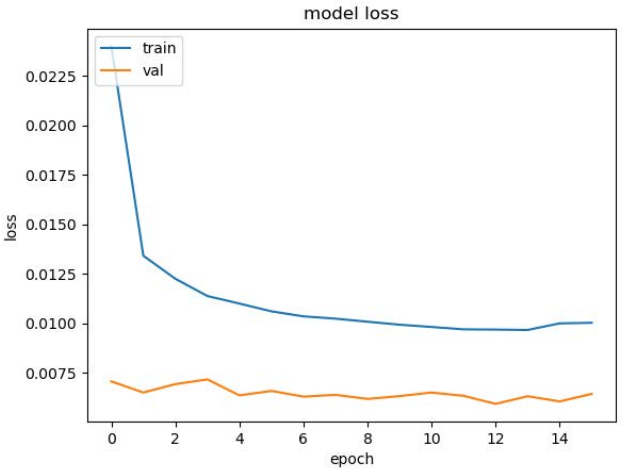
In the case of AWID3, the MLP model achieved the best average AUC score of 96.47%. The fastest DNN model was Denoising stacked Autoencoders, with ≈40 hours of training time. Epochs in AWID3 DNN analysis were quite high, i.e., having in most cases 6 to 16 epochs in best folds. Figure 1 shows the MLP (best performer) model accuracy graph, in the best/worst fold case. Recall that the difference between validation and training accuracy is due to the Dropout effect.

By referring to the bottom-most line of Tables 7 and 8, it is observed that, opposite to shallow analysis, the difference in terms of the average AUC and F1 scores between the two datasets are substantially larger; approximately 15% and 18%, respectively. As detailed in section VI, it can be assumed that the number of samples was insufficient to train

the DNN models at a superior level. Overall, it is expected that better and more converging average detection scores will be achieved with a higher number of training samples and possibly the use of other DNN models; however, this supposition is left to be verified by future work.



Best fold case



Worst fold case

FIGURE 1. MLP model best/worst case of loss for AWID3.

C. ADDITIONAL FEATURES

Following the discussion presented in section III-A and the results given in section IV, a basic question may arise: are there any additional generic PHY or MAC features that may assist further the ML algorithms or models in producing better results? Based on empirical observations, we can suggest the below three features that satisfy all the criteria of subsection III-A, but are specific to AWID3. This is because all these features were introduced in Wireshark v2, so they were not available at the time AWID2 was built.

1) *wlan_radio.phy*: It designates the 802.11 protocol used. For instance, the 802.11n has a different value from 802.11ac. If the network uses a contemporary wireless protocol, e.g., 802.11ac or 802.11ax, the current feature may assist in recognizing an attacker who uses an outdated equipment, e.g., 802.11n. Put differently, this information can assist the model in detecting flooding and impersonation attacks. Note that the current feature is different from the *radiotap.datarate/wlan_radio.data_rate* one, since the latter can specifically (uniquely) pinpoint an attacker.
2) *wlan_radio.signal_dbm*: It denotes the signal strength of the transmitting device. If used in combination with the *radiotap.dbm_antsignal* may assist in detecting flooding and impersonation attacks.
3) *wlan_radio.duration*: It is similar to the *wlan.duration*, and in combination with the latter, may contribute in ameliorating the detection accuracy.

TABLE 7. Results of DNN model analysis for AWID2. AE stands for autoencoder model.

Model	AUC	Prec	Recall	F1	Acc	Epochs	T.E.Time
MLP	81.79	95.44	75.50	81.98	97.75	8.2	20:52:47
AE	80.07	86.54	73.34	76.04	97.27	6.4	22:16:38
AVG	80.93	–	–	79.01	–	–	–

TABLE 8. Results of DNN model analysis for AWID3. AE stands for autoencoder model.

Model	AUC	Prec	Recall	F1	Acc	Epochs	T.E.Time
MLP	96.47	99.65	95.68	97.55	99.73	19.8	49:55:02
AE	95.39	99.64	94.31	96.78	99.66	12.1	40:28:05
AVG	95.93	–	–	97.16	–	–	–

To verify if the aforementioned assumption holds, that is, more generic features can improve the detection metrics, we amended the AWID3 dataset including the above three features and re-executed the experiments only for the best performers for both the swallow and DNN types of analysis. The parameters values were the same as in Tables 3 and 6.

As shown in Table 9, this new experiment demonstrated that the ET achieved slightly worse results (-0.25%) in terms of average AUC as compared to the use of 16 features, and with $\approx$4 hours more execution time. The confusion matrix is presented in section VI.

Regarding the MLP model, as shown in Table 9, an increased average AUC score of $+0.76\%$ was observed, with the best fold reaching an AUC score of 99.05% and an improved F1 score by $+0.42\%$. Interestingly, the total execution time (T.E. Time) was $\approx$4 hours less than that of the 16 features case. This shorter time roots in the total number of epochs, each type of analysis needed. That is, in the case of the 16 features, the model required a total of 198 epochs, or 19.8 mean epochs per fold. While during the analysis of the 19 features, the model required a total number of 184 epochs, resulting in 18.4 mean epochs per fold. Therefore, through the acquisition of more information due to the three additional features, the MLP model was trained faster ($\approx$7% lesser epochs), thus leading to an improved total execution time by $\approx$7%. The confusion matrix in section VI contains the results of the average confusion matrices. Figure 2 illustrates

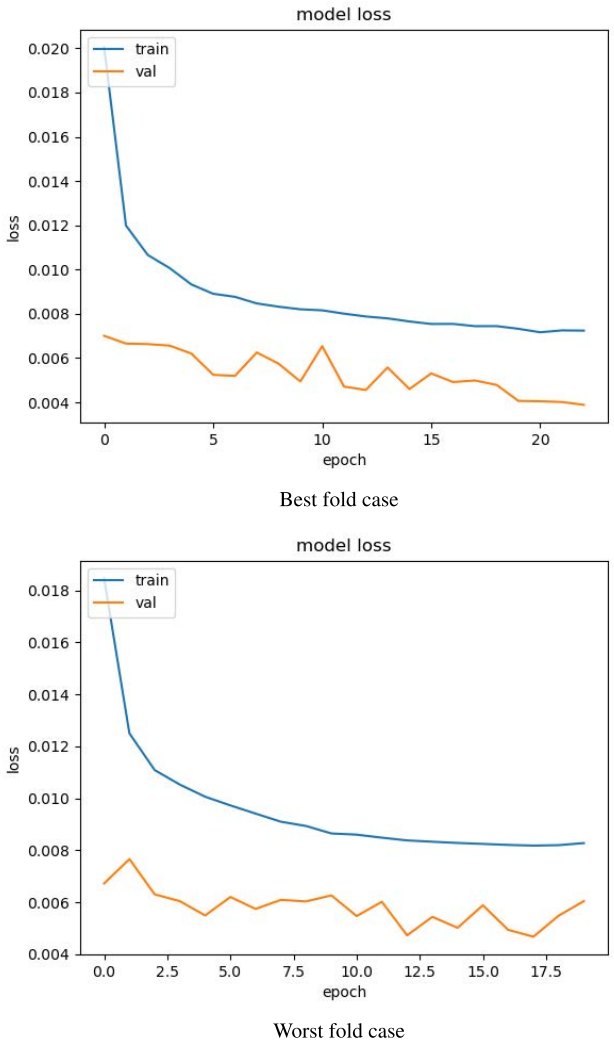


FIGURE 2. MLP model best/worst case of loss for AWID3 with 16 + 3 features.

the comparison between the training and validation accuracy of MLP model. Altogether, the answer to the question posed at the start of this subsection is that the increased number of cherry-picked features can aid DNN models in improving the detection rate.

TABLE 9. AWID3 results of ET and MLP models analysis using the 16 + 3 features. N/A: not applicable.

Model	AUC	Prec	Recall	F1	Acc	Epochs	T.E.Time
ET	99.24	99.76	98.98	99.37	99.94	N/A	15:18:43
MLP	97.23	99.49	96.61	97.97	99.77	18.4	45:20:40

D. REDUCED SET OF FEATURES

Similar to subsection IV-C, new questions may arise: Do the 16 features selected achieve optimal predictive accuracy, or perhaps a smaller set achieves better results? Moreover, which of the 16 features are more significant for both the datasets? And can this possibly reduced set of most significant features accomplish results that are close (less than 2% in terms of the AUC score) to those of the full 16-set? Towards providing an answer to the previous questions, the following empirical observations can be made.

With reference to an 802.11 IDS, typically, numerical data contain more information compared to categorical ones. Based on the feature selection process given in subsection III-A, only four features, namely *frame.len*, *radiotap.length*, *radiotap.dbm_antsignal*, and *wlan.duration*, are numerical. From them, the *radiotap.length* may be omitted, since it is contained in the *frame.len*.

From the remaining 12, some are too generic and probably may either contain less useful information (this is the case with the radiotap ones) or the same information may be present in another feature. The latter case pertains to the *wlan.fc.type* and *wlan.fc.subtype* features, which however can be indirectly deduced from the *frame.len*. For instance, taking as example the beacon and QoS data frames, the following observations can be inferred. Both are of 1000 subtype, with the former to be a management (01) and the latter a data (10) frame. With reference to AWID3, the length of these frames are 342 and 260 bytes, respectively.

From the remaining 6 features, we could consider taking on board the *wlan.fc.ds*, *wlan.fc.frag*, and *wlan.fc.protected*. The remaining three (*wlan.fc.retry*, *wlan.fc.pwrmgt*, and *wlan.fc.moredata*) are only used in specific cases. For instance, the *wlan.fc.pwrmgt* is set by a STA to inform the AP that it transits to power saving mode. However, this is done with a single frame and the STA will not set the same flag, but possibly only after the AP wakes it up.

To confirm the aforesaid empirical observations, we resorted to feature importance. The latter is usually calculated based on two methods: (a) mean decrease in impurity, and (b) feature permutation. While both methods are valid, the first method can lead to misleading results in case there exist features with many unique values. For this reason, we chose feature permutation implemented on two tree-based classifiers; LightGBM, which presented the best results in terms of the F1 score for both the datasets, and RF, which had the worst results amongst the 3 tree-based classifiers for the same metric. Note that with reference to Tables 4, 5 and 7, 8 it can be said that, in comparison to Stochastic, Linear, and NN models, Tree algorithms presented overall better results on both the datasets. In this respect, feature importance was only considered for Tree-based algorithms.

Feature importance considered the same classifiers' setup parameters as in Table 3, and each dataset was analyzed following the data preprocessing steps given in subsection III-B. We divide each dataset with a stratified split to 60 and 40 for the training and testing sets, respectively. Then, the testing set was evaluated 10 times (n_repeats), for AWID2 and AWID3, while both the datasets had the full 16-features set. After omitting features with less than 2% importance, the results are illustrated in Figures 3 and 4 for LightGBM and Random Trees, respectively.

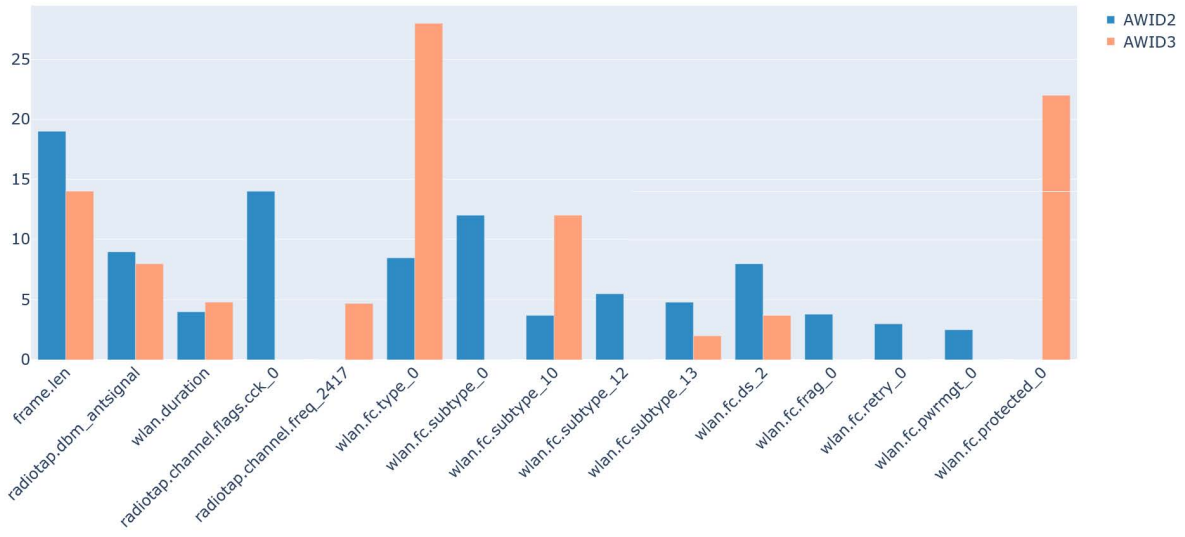


FIGURE 3. LightGBM permutation feature importance evaluation on the 16 features of AWID2 and AWID3. The Y axis represents a percentage.

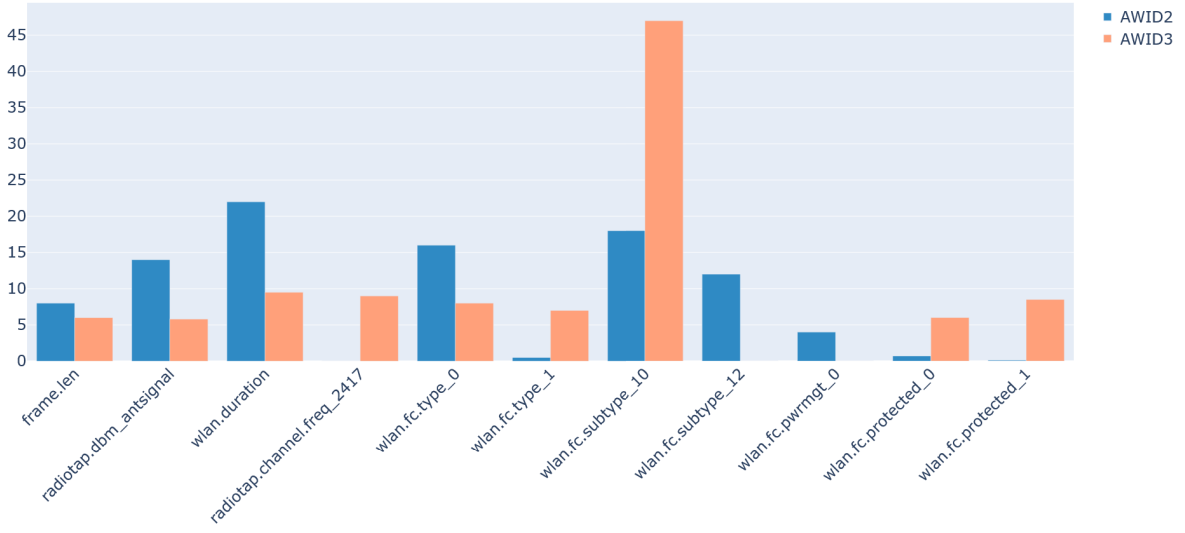


FIGURE 4. RF permutation feature importance evaluation on the 16 features of AWID2 and AWID3. The Y axis represents a percentage.

TABLE 10. AWID2 Tree-based results with feature sets 1 and 2.

	Set 1 (subtype)						Set 2 (type)					
Model	AUC	Prec	Recall	F1	Acc	T.E.Time	AUC	Prec	Recall	F1	Acc	T.E.Time
LightGBM	89.07	97.50	85.38	90.77	98.68	00:08:48	88.55	96.56	84.70	89.98	98.56	00:07:33
DT	92.61	95.52	89.94	92.59	98.93	00:01:31	88.81	95.04	84.78	89.32	98.52	00:00:46
RF	89.11	97.50	85.37	90.71	98.69	00:36:53	89.22	96.17	85.50	90.28	98.63	00:32:02
ET	95.03	97.22	93.33	95.19	99.26	01:57:10	93.28	96.14	90.95	93.38	99.02	01:13:58

The X axis in the figures refers to each column importance. For example, as expected, the *wlan.fc.type* feature contains three values, namely, 0, 1, and 2. When the OHE applies to the feature, the *wlan.fc.type* is deleted, and three new are created, i.e., *wlan.fc.type_0*, *wlan.fc.type_1*, and *wlan.fc.type_2*, with each one referring to the respected value. So, with reference to the LightGBM evaluation, Figure 3 shows that the *wlan.fc.type* feature is important for both the datasets, but

TABLE 11. AWID2 Tree-based results with feature sets of 3 and 4.

	Set 3 (DS)						Set 4 (protected)					
Model	AUC	Prec	Recall	F1	Acc	T.E.Time	AUC	Prec	Recall	F1	Acc	T.E.Time
LightGBM	88.36	96.18	84.40	89.64	98.53	00:07:44	88.36	96.18	84.40	89.64	98.53	00:07:00
DT	88.79	94.47	84.77	89.10	98.49	00:00:41	88.79	94.47	84.77	89.10	98.49	00:00:36
RF	88.94	96.49	85.14	90.19	98.62	00:30:39	88.89	96.66	85.07	90.21	98.63	00:30:38
ET	91.69	96.29	88.79	92.16	98.87	01:01:22	91.67	96.30	88.75	92.14	98.87	00:40:48

TABLE 12. AWID3 Tree-based results with feature sets 1 and 2.

	Set 1 (subtype)						Set 2 (type)					
Model	AUC	Prec	Recall	F1	Acc	T.E.Time	AUC	Prec	Recall	F1	Acc	T.E.Time
LightGBM	86.39	99.65	79.74	85.59	99.35	00:49:31	85.85	99.46	79.12	85.22	99.30	00:56:29
DT	94.62	95.30	91.95	93.54	99.59	00:11:47	95.46	92.40	93.39	92.87	99.53	00:08:31
RF	84.72	99.58	77.47	83.71	99.24	04:12:45	85.12	99.64	77.96	84.01	99.28	04:04:07
ET	74.87	76.04	62.74	65.97	98.67	12:06:05	79.17	99.37	68.80	70.48	99.02	05:37:14

TABLE 13. AWID3 Tree-based results with feature sets 3 and 4.

	Set 3 (DS)						Set 4 (protected)					
Model	AUC	Prec	Recall	F1	Acc	T.E.Time	AUC	Prec	Recall	F1	Acc	T.E.Time
LightGBM	85.62	99.63	78.87	85.26	99.28	01:03:37	94.47	99.20	91.90	95.18	99.69	00:47:11
DT	91.09	94.51	87.70	90.04	99.46	00:09:00	97.16	94.72	95.67	95.19	99.69	00:06:22
RF	85.42	99.70	78.58	84.96	99.28	04:17:44	91.26	99.30	87.27	92.17	99.53	05:13:36
ET	74.17	99.49	63.93	71.66	98.55	06:56:45	89.98	99.24	85.46	91.00	99.47	05:20:32

only for the 0 value (wlan.fc.type_0). This means that the other two columns, i.e., *wlan.fc.type_1* and *wlan.fc.type_2* add noise to the analysis of the classifier, with the true importance of this feature to be possibly lower, ≈1/3 of 27% or 9% for AWID3. This is relevant to every OHE feature but the numerical ones, which were converted using the Min-Max scaling technique.

Overall, from figures 3 and 4 it is observed that across both the datasets, three features stand out: *frame.len*, *radiotap.dbm_antsignal*, and *wlan.duration*. Additionally, *wlan.fc.type*, *wlan.fc.subtype*, *wlan.fc.ds* seem to present significant importance for both the datasets, while, as expected, *wlan.fc.protected* has high importance only in AWID3.

The results of the feature importance analysis largely corroborate the empirical analysis done earlier in the first paragraphs of the current subsection. Exceptions are the *wlan.fc.type* and *wlan.fc.subtype* features; it seems that, at least based on tree-based models, the relevant information cannot be extracted from the *frame.len* feature. Another exception pertains to the *radiotap.channel.freq* for AWID3. Namely, although the *radiotap.channel.freq* presented ≈5 to 8% feature importance in both LightGBM and RF for AWID3, this was due to the Krack impersonation attack, which utilizes the 5GHz radio channel along with the 2.4GHz one.

To further validate our analysis, we tested a quartet of feature sets. Since the feature importance evaluation was done on tree-based classifiers, we utilized only the tree-based models. Based on the empirical analysis, we kept only four features per set. All the assessed sets given below contain the three numerical features, i.e., the most informative ones, plus one different OHE feature per set.

- *Set 1 (subtype)*: *frame.len*, *radiotap.dbm_antsignal*, *wlan.duration*, and *wlan.fc.subtype*
- *Set 2 (type)*: *frame.len*, *radiotap.dbm_antsignal*, *wlan.duration*, and *wlan.fc.type*
- *Set 3 (ds)*: *frame.len*, *radiotap.dbm_antsignal*, *wlan.duration*, and *wlan.fc.ds*
- *Set 4 (protected)*: *frame.len*, *radiotap.dbm_antsignal*, *wlan.duration*, and *wlan.fc.protected*

With reference to Tables 10 and 11 for AWID2, and 12 and 13 for AWID3, the following general observations per dataset can be made. First, the minimum number of features seems to be different for each dataset, 4 for AWID2 and more than 4 for AWID3. Second, the analysis regarding the least number of features pertains to Tree-based algorithms; preliminary experiments done over the remaining ML models with the same parameters indicated that they require more features to perform well.

Regarding AWID2, the best AUC score achieved for Set 1 (subtype) by the ET classifier was almost the same (−0.13%) with the best performer (DT) for the same metric reported in subsection IV-A2. Notably, the best performer in terms of both the AUC and F1 metrics across all the four sets was the ET classifier. Set 2 (type) yielded the second-best score with an AUC of 93.28% (−1.88%). Sets 3 (ds) and 4 (protected) followed with almost the same prediction rate, i.e., 91.69% (−3.47%) and 91.67% (−3.49%), all vis-à-vis the best performer (DT) over the 16-features set.

For AWID3, the best performer across was all the four sets was the DT model, scoring an AUC of 97.16% at best; recall that according to subsection IV-A2, the ET classifier has shown superior performance with an AUC score of 99.49%. As expected, the best set was the fourth one (protected) yielding an AUC score of 97.16% (−2.33%). The rest of the sets, namely Set 2 (type), Set 1 (subtype), and Set 3 (DS), demonstrated a lower prediction rate, i.e., 95.46% (−4.03%), 94.62% (−4.87%), and 91.09% (−8.4%), respectively. Again, the number in parentheses are in comparison to the ET classifier over the 16-features set, as shown in Table 5.

V. FEATURE TRANSFERABILITY

Following the analysis done in section III-A regarding the selection of features and the results given in the previous subsections, the current subsection attempts to answer another key question: Are the selected set of 16 features directly transferable across datasets? To respond to this matter, we compared the efficiency of the 16-features set against other sets of features of variable sizes. More specifically, we first performed an analysis on AWID2 by selecting a much larger set of 30 features (referred to as "30F" in the following), i.e., the 16 features of Table 2 plus the 14 commonest ones in the related work from Table 1: *frame.time_epoch* to *frame.time_relative*, *radiotap.mactime*, *radiotap.datarate*, *wlan.ra* to *wlan.sa*, *wlan.seq*, *wlan_mgt.fixed.reason_code*, *wlan_mgt.fixed.beacon*, and *data.len*. Note that some of the aforementioned features may have a different name in AWID3 because they were drawn from a newer version of Wireshark.

For this experiment, only the best performers of subsection IV-A, namely ET, DT, and LightGBM were utilized. All the features have been converted through the Min-Max scaling technique, given that it is prevalent in the AWID2 literature, even for categorical type of data. The top-most sub-table of Table 14 contains the results of this analysis using 10-fold stratified cross validation. By comparing the results of this sub-table against Table 4, DT scored a better AUC by ≈0.5%, while LightGBM resulted in a much greater AUC score, i.e., 4.17%. It is therefore obvious that the 30F analysis yields superior results vis-à-vis the 16 features set. However, as discussed in section III-A, certain features can add a bias effect, leading to overfitting.

To further support this contention, we examined an indicative case by training the classifiers in AWID2 and testing them in AWID3; the training set was the *AWID2-CLS-R* with only the Normal and Flooding classes, while the test set was the AWID3 *Deauth.pcap* containing only Normal and Deauthentication traffic. Different sets containing 30 (30F), 27 (27F), 13 (13F), and 5 (5F) features were examined as shown in the four bottom-most sub-tables of Table 14. The results clearly confirm our empirical analysis of section III-A that the 30F set (and generally any arbitrarily selected large set of features) is not transferable, achieving a low AUC score of 50% at best; characteristically, the confusion matrix of this analysis classified all the samples as Normal traffic.

To investigate further this outcome, we realized that the *radiotap.channel.freq*, *radiotap.flags.type.cck*, and *radiotap.flags.type.ofdm* features should be removed because AWID2 and AWID3 were captured on a different radio channel, and the two flag-based features do not offer any useful information for a flooding attack. Additionally, less than 100 rows which contained unrelated data between the two datasets, namely *wlan.fc.subtype_14*, *wlan.fc.subtype_15*, and *wlan.fc.ds_3* were removed as well. Nevertheless, as shown in Table 14, the results on the remaining 27F set were very similar to the 30F analysis, thus further corroborating our empirical analysis.

On the other hand, an additional analysis using the 13F set (the 16 features set of subsection III-A minus the above-mentioned three) yielded promising results for two out of the three tree-based classifiers. This outcome strongly suggests that the feature selection process of subsection III-A is sound, making the features transferable between different 802.11 datasets; however, researchers should take special care for potentially incompatible features due to diverse versions of the standard, or features that are irrelevant to the specific attacks the IDS is tasked to identify depending on the case.

With reference to IV-D, a last experiment was conducted with a 5F set, that is, the features in *Set 1* and *Set 2* of subsection IV-D, namely *frame.len*, *radiotap.dbm_antisignal*, *wlan.duration*, *wlan.fc.type*, and *wlan.fc.subtype*. As presented in the bottom-most sub-table of Table 10, one of the classifiers demonstrated similar results to the 13F case.

VI. DISCUSSION

Following the experiments and observations of the previous sections, the current one provides a more in-depth discussion of the findings, and compares the derived results with those given in the relevant literature.

A. SHALLOW CLASSIFICATION

As given in Figures 4 and 5, LightGBM achieved the best F1 score on both the datasets; 95.36% and 99.55% for AWID2 and AWID3, respectively. On the other hand, the SGDClassifier presented the worst F1 score, i.e., 84.56% and 96.60% for AWID2 and AWID3, respectively. Overall, regarding AWID2, the results are in the same range irrespective of the analysis type, i.e., an around 4 to 10% difference between the best and the worst average AUC: 95.16% and 85.99% for shallow classifiers, and 84.91% and 80.97% for DNN. The same result is also observed for the same dataset regarding the F1 score: between 95.36% and 84.56% for shallow and 87.49% and 76.04% for DNN. For AWID3, the disparities observed in the results for the same metrics are much smaller, namely around 1 to 4% for AUC: 99.42% and 95.17% for shallow classification and 96.47%/95.54% for DNN. The same picture is true for the F1 metric: 99.55% and 96.60% for shallow and 97.55% and 96.55% for DNN.

The tree-based algorithms demonstrated higher detection metrics in terms of AUC and F1 in comparison to the other two categories, namely Stochastic and Linear-based.

TABLE 14. Results on feature transferability. The analysis done on the 30F set used 3 classes, namely Normal, Flooding, and Impersonation. The remaining tests were made with two classes, Normal and flooding. N/F and N/D stand for normal/flooding and normal/deauth, respectively.

Model	AUC	Prec	Recall	F1	Acc	T.E.Time
AWID2 10-fold stratified cross val (30F) with 3 classes						
ET	97.20	99.43	96.24	97.78	99.66	01:20:39
DT	95.64	97.91	93.99	95.88	99.41	00:06:12
LightGBM	98.58	99.73	98.10	98.90	99.83	00:10:20
Training with AWID2 (N/F) and testing with AWID3 (N/D) (30F)						
ET	50.00	48.80	50.00	49.39	97.61	00:06:55
DT	49.81	48.80	49.81	49.30	97.23	00:00:40
LightGBM	50.00	48.80	50.00	49.39	97.61	00:02:24
Training with AWID2 (N/F) and testing with AWID3 (N/D) (27F)						
ET	50.00	32.54	33.33	32.93	97.61	00:08:37
DT	49.81	48.80	49.81	49.30	97.23	00:00:30
LightGBM	50.00	48.80	50.00	49.39	97.61	00:02:06
Training with AWID2 (N/F) and testing with AWID3 (N/D) (13F)						
ET	95.14	99.80	95.14	97.35	99.76	00:04:18
DT	93.48	98.69	93.48	95.93	99.63	00:00:10
LightGBM	57.85	98.50	57.85	63.03	97.92	00:00:31
Training with AWID2 (N/F) and testing with AWID3 (N/D) (5F)						
ET	65.01	98.82	65.01	72.61	98.27	00:05:25
DT	93.41	98.69	93.41	95.89	99.63	00:00:12
LightGBM	57.81	98.39	57.81	62.96	97.92	00:00:37

Precisely, two out of three tree-based algorithms yielded an at least ≈4% better F1 score for AWID2 vis-à-vis the other two categories of classifiers. Regarding the training time, AWID2 showed to be ≈5 times faster in comparison to AWID3. This is however expected since AWID3 is at least five times bigger in terms of samples, as compared to AWID2.

Both the datasets performed better in shallow classification analysis. This is especially obvious for AWID2 that scored an ≈8% worse F1 average score in DNN analysis, as compared to that of shallow classification. On the other hand, for AWID2, the MLP model yielded a higher AUC average score of 81.79% as compared to the other one. The same behavior was observed for DNN analysis on AWID3, i.e., MLP model performed better than Autoencoder: 96.47% (MLP) vs 95.39% (Autoencoders).

With reference to the confusion matrices in Figure 5, AWID2 presented superior results with DT. The latter misclassified ≈738 (0.3%), ≈651 (11%), and ≈639 (9%) samples of the Normal, Flooding, and Impersonation classes, respectively. For the same dataset, the best performer regarding the Normal class was the RF, which misplaced about 81 samples (0.03%). The Impersonation class suggested the LightGBM as the best performer; ≈539 (7%) samples were misclassified. Generally, 5 out of the 7 shallow classifiers managed to predict correctly at least the 72% and 77% of the Flooding and Impersonation classes, respectively. For easy reference, Figure 5 depicts the remaining ML model confusion matrices.

For AWID3, the confusion matrices in Figure 6 demonstrated that LightGBM missed only ≈53 (0.003%) samples of the Normal class. Generally, regarding the latter class, all the classifiers missed less than 150 (0.01%) samples, which is a rather excellent result. A similar behavior was observed for the Flooding class, where all classifiers but one (SGDClassifier) managed to miss less than 1,500 or 4% of the samples. Put differently, the great majority of the classifiers managed to predict correctly the 96% of the Flooding class. Also on the positive side, as shown in Figure 6, ET, which was the best performer, misplaced ≈173 or 0.5% of the samples from the Flooding class. Last but not least, as illustrated in Figure 6, the Impersonation class also presented good results, with 6 out of 7 classifiers to miss less than ≈565 or 3% of the samples. This translates to a prediction accuracy of 97%. On the negative side, SGDClassifier and RF misplaced 4500 (14%) and 2005 (12%) samples from the Flooding and Impersonation classes, respectively.

B. DNN

For DNN analysis, and with reference to Figure 7, the best performer for AWID2 was MLP. This model misplaced only 77 (0.03%) samples of the Normal class, while missed only ≈680 or 0.3% of the samples of the same class. The Denoising stacked Autoencoder presented the worst prediction percentage for the Normal class, i.e., more than 1,300 or 0.6% misplaced samples. Regarding the Flooding class, almost half of the samples were misplaced, with the best performer (MLP) to predict correctly only 2,935 samples, i.e., 51% of the total samples of this class. The Impersonation class presented better results; MLP misplaced ≈ 1,715 (24%) samples.The Denoising stacked Autoencoder showed significantly poorer results, missing around 2,245 (32%) samples.

As with shallow classification, AWID3 presented significantly better results in DNN analysis. Specifically, with reference to Figure 8, all DNN models excelled in the Normal class, misplacing only about 120 or 0.008% of the samples. For the Flooding class, only the MLP managed to achieve better results, i.e., it managed to identify correctly 700 or 2% more samples than the rest of the DNN models which missed ≈4,500 or 14% of the samples. Regarding the Impersonation class, all DNN models presented very good results, i.e., misplacing less than 560 (3.5%) samples each.

C. ADDITIONAL FEATURES

With reference to the increased feature set given in section IV-C, and by observing the confusion matrices in Figure 9, it can be said that ET misclassified a rather small number of samples, less than 800 samples altogether. Moreover, MLP demonstrated better results for the Flooding class, with ≈1,000 (3%) more samples being classified correctly vis-à-vis the 16-features set. Nevertheless, the Normal and Impersonation classes showed small negative or positive differences compared to the 16-features set; the algorithm missed ≈150 (0.01%) more samples in the Normal class and predicted correctly ≈50 (0.3%) more samples in the Impersonation one. Overall, for shallow classification, the additional three features do not seem to improve the results: in terms of both the AUC and F1 metrics, the results were a bit worse, namely 99.49% vs. 99.24% (−0.25) and 99.52% vs. 99.37% (−0.15) for the 16- and 19-features set, respectively.

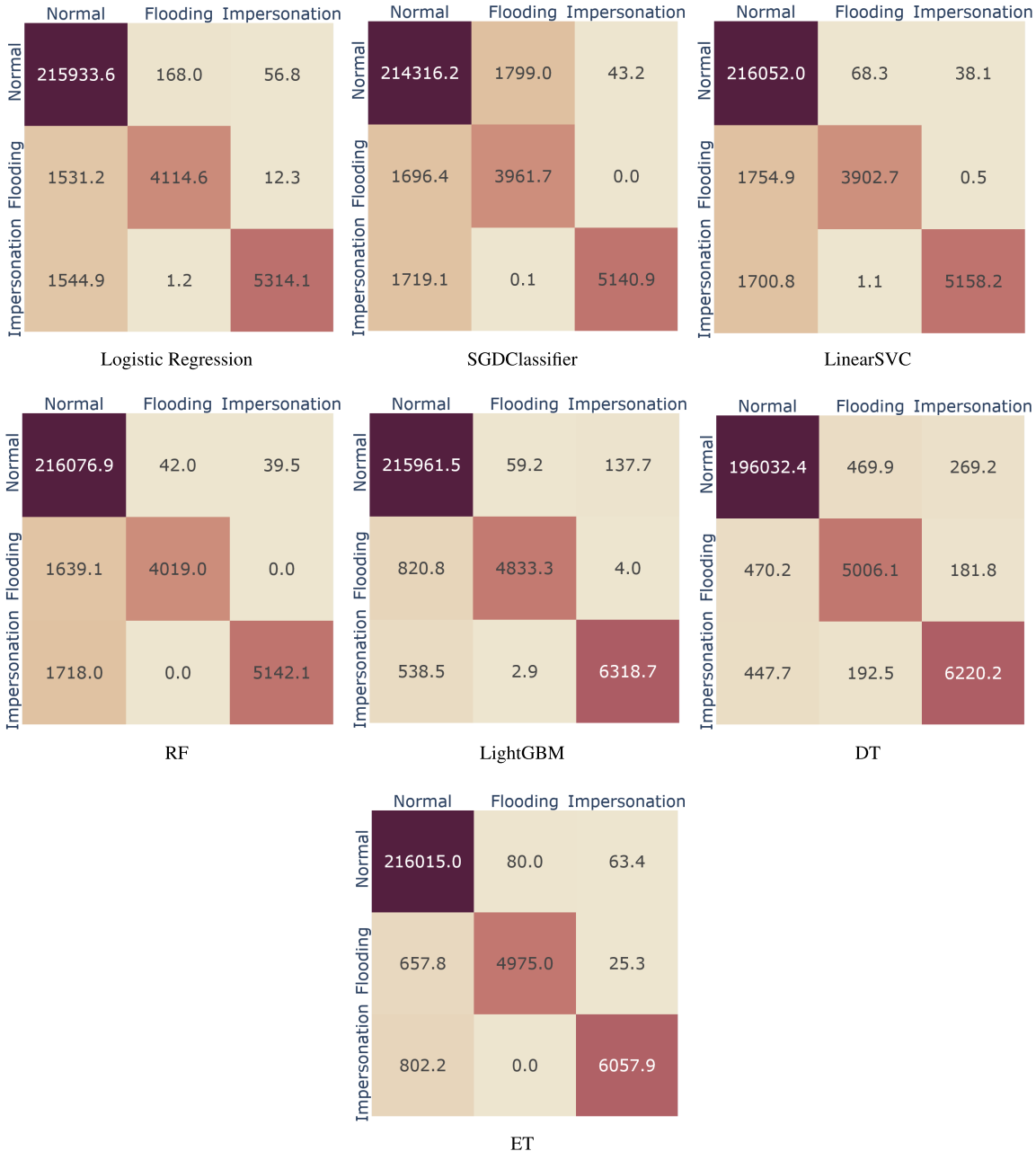


FIGURE 5. Confusion matrices for AWID2. Average results over all folds for shallow model analysis.

However, this outcome is reversed for the MLP model: 96.47% vs. 97.23% (+0.76) and 97.55% vs. 97.97% (+0.42).

D. REDUCED SET OF FEATURES

With reference to Figure 10, using the ET classifier along with features Set 1 (subtype) on AWID2 yielded ≈114 or 0.05% instances of the Normal class misclassified as Flooding. Moreover, ≈341 or 0.15% of the Normal samples were recognized as Impersonation. Similar was the behavior for the other two classes, namely Flooding and Impersonation. The former class misplaced ≈617 or 10% as Normal samples, and ≈19 or 0.33% as Impersonation samples. The Impersonation class, showed a prediction rate of 8% (≈579) misplaced samples as Normal ones, while, less than 0.07% or ≈5 samples were wrongly classified as Flooding instead of Impersonation.

Since the best AUC score of Set 1 (subtype) for AWID2 is almost equal to that of the 16-features set, future work may further assess if these quartets of high-importance features can be combined with others, including *radiotap.channel.flags.cck*, *wlan.fc.frag*, *wlan.fc.retry*, and *wlan.fc.pwrmgt*, towards augmenting the prediction rate of classifiers. That is, with reference to Figures 3 and 4, the

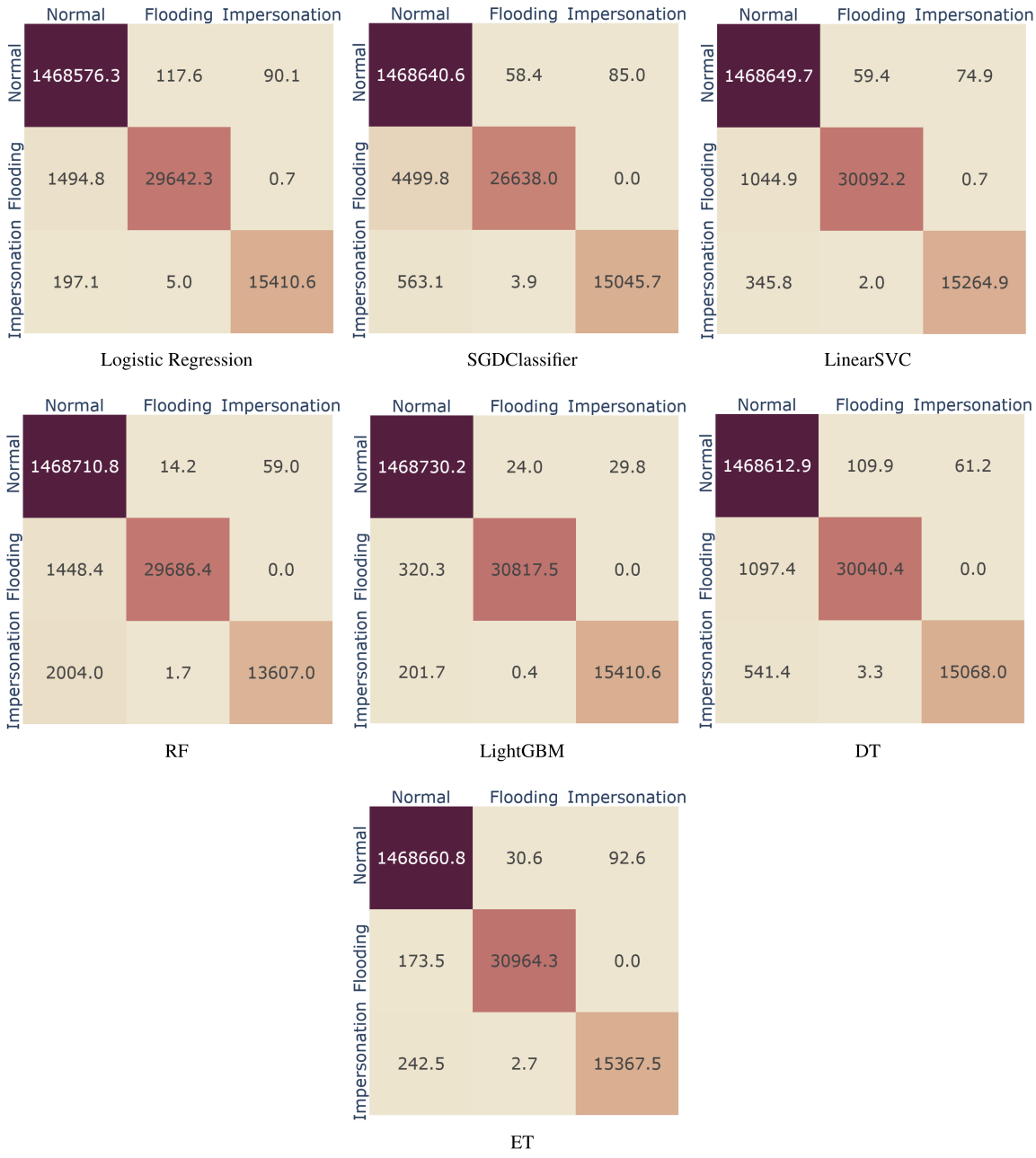


FIGURE 6. **Confusion matrices for AWID3. Average results over all folds for Shallow model analysis.**

aforesaid four OHE features showed some, even minor, significance. Additionally, some of them, e.g., *wlan.fc.retry_1*, seem to also add noise to the prediction rate. Naturally, future work may also focus on additional set of features and other ML models, including Logistic Regression.

Regarding AWID3, the best results obtained on the reduced set 4 (protected) showed a negative difference of −2.33% in terms of the AUC metric in comparison to the best performer in Table IV-A2. This reduced detection performance is somewhat expected as AWID3 contains more advanced attacks, and therefore the classification process needs more information to reach a decision. For this reason, the confusion matrix of DT with Set 4 (protected) suggests that the algorithms faced difficulties in classifying different kinds of Impersonation attacks, especially instances of the Krack one. For instance, as illustrated in Figure 10, the DT classifier over Set 4 (protected), misclassified ≈1,677 or 10% of the samples as Normal, instead of Impersonation. Moreover, the same algorithm over the same 4-features set, misclassified ≈660 or 0.04% and 1,684 or 0.11% of the Normal samples as Flooding and Impersonation, respectively. On the bright side, Flooding samples only misplaced ≈133 or 0.42% of the samples as Normal and ≈500 or 1% as Impersonation. In light of these results, it can be said that AWID3 requires

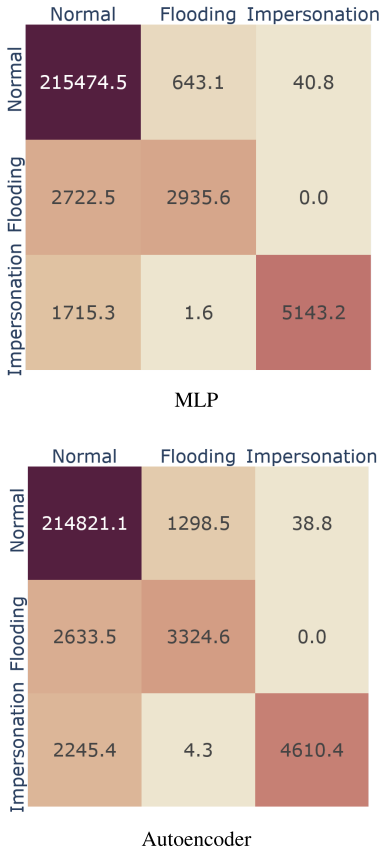


FIGURE 7. Confusion matrices average results across all folds for AWID2 DNN model analysis.

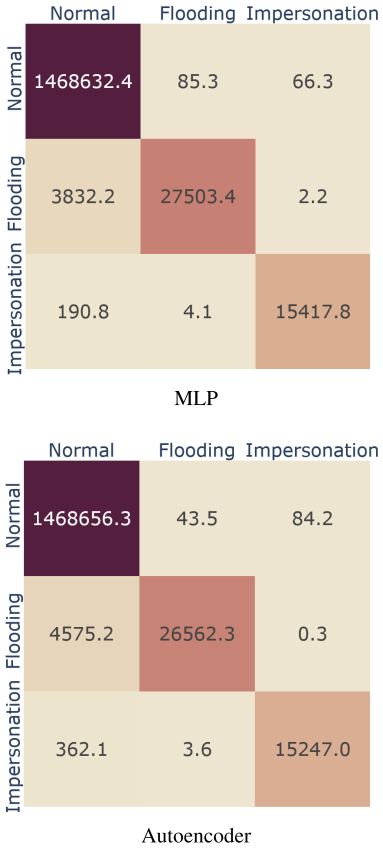


FIGURE 8. Confusion matrices average results across all folds for AWID3 DNN model analysis.

more information to discern Impersonation attacks. A clear direction for future work is to combine the features of Set 4 (protected) with some or all of the remaining ones, namely *radiotap.channel.freq*, *wlan.fc.type*, and *wlan.fc.subtype*.

E. COMPARISON WITH PREVIOUS WORK

This section complements section II by providing deeper comparisons with the related work. Table 15 gathers the common characteristics, including methodology and results, of major past works considering either AWID2 or AWID3. Additionally, the table includes the best performers per dataset as seen by this work. The contribution [14] mentioned in section II is omitted because it focused on performance issues (CPU vs. GPU), rather than intrusion detection.

While significant effort has been devoted in providing common characteristics across all the works included in the Table, the reader should consider this comparison as "loose." This is because there is a lot of – often not detailed or missing – processing done to the datasets considered per work, including the way the ML models were configured and trained. Given that all the works provide an accuracy score, the rows of the table are sorted based on this metric. Even more, as all but one of the previous AWID2 works consider four classes (while the current takes into account three classes), we have rerun our experiments only for the best performers using four classes. The respective results are shown using the "◇" symbol in the table. Recall that the cardinal reason why this work used three classes instead of four is for achieving compatibility between AWID2 and AWID3, which consider 4 and 3 classes, respectively. In any case, as explained below, the fourth class (Injection) can be easily discerned by the classifiers.

As observed from the Table, most works do not report important evaluation metrics and omit details regarding the methodology used. Moreover, only three of the included works employed k-fold cross validation; recall that k-fold can alleviate overfitting and produce more robust and accurate prediction scores. Additionally, some contributions mention evaluation metric scores that seem questionable. For instance, each one of the works in [5], [9], [13], AND [12] presents an almost equal Precision, Recall, F1, and Accuracy scores. Based on these results, it can be assumed that these works most probably employ a balanced dataset, and indeed this is the case with [5]. Nevertheless, in real-world scenarios, the wireless traffic will be, in all likelihood, imbalanced. Namely, it is highly unlikely that the number of attack and normal frames will be equal, and in some cases, the attack frames in 802.11 networks can be much more than the Normal ones; think for instance a deauthentication or disassociation kind of DoS attack.

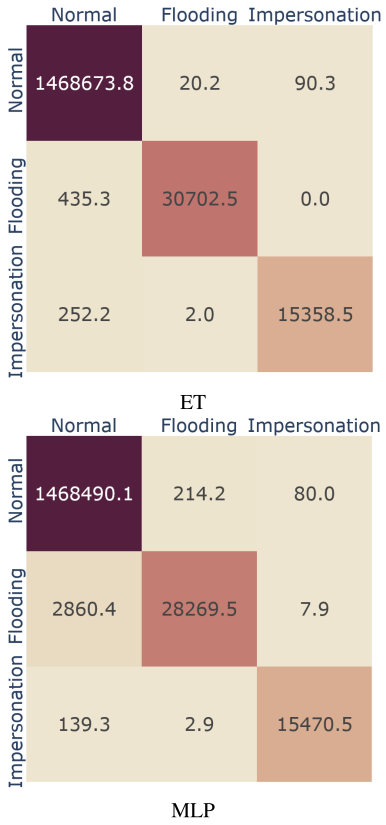


FIGURE 9. Confusion matrices for AWID3 (16 + 3 features set). Average results over all folds.

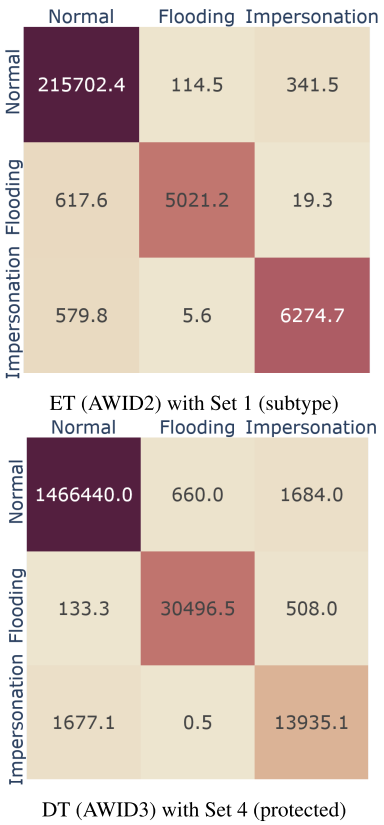


FIGURE 10. Confusion matrices average results across all folds of the best performers for AWID2 (ET) and AWID3 (DT) model analysis with 4 features of Set 1 (subtype) and Set 4 (protected), respectively.

Moreover, as discussed in section III, many features used by the literature are highly susceptible to lead to overfitting, rendering the IDS ineffective for real-life 802.11 networks. For instance, as already shown in Table 1, if the IDS is trained using MAC address based features, as it is the case with most of the works in Table 15, it will most probably suffer from overfitting issues.

Most of the works do not use any regularization method. However, regularization is essential for providing – at least up to some level – guarantees that the predictions are accurate and with minimal overfitting. Other defects pertain to the employed data preprocessing techniques, where some works employ improper methods. For instance, transforming the *wlan.ta* feature to numerical values (this is probably done because following the right way of transforming categorical data with OHE is resource consuming, e.g., it yields more than 10K columns with AWID2), it makes the predictions questionable. It is emphasized that having good, or even superior results, does not necessarily mean that the generated IDS model is sound and will behave the same way if fed with a different test set.

As already explained in subsection III-A, an additional issue that can negatively affect the prediction rate of a classifier is the conversion of intrinsically time-series features with Min-Max scaling; this assertion is also supported by the results of subsection V regarding the 30F/27F set. Namely, such a conversion alters the origin of each time-series feature into indistinct values. Further, the zeroing of non-existent (“?”) or blank values, or its replacement with artificially ones, should be done with due care for the former case, and avoided for the latter. For instance, zeroing blank or “?” values, may alter the prediction rate of, say, tree-based algorithms, because they would possibly consider this value in their predictions, leading to disputable results.

From the Table 15, it is obvious that the Accuracy metric is heavily exploited by previous work for assessing the IDS model. This is also due to the use of balanced datasets, as explained earlier in this section. Nevertheless, leaving aside the argumentation about the use of a balanced vs. imbalanced dataset, this metric should be always weighted in conjunction with other results. For instance, although the contributions in [7] AND [8] report a high accuracy of 95.87% and 99.42%, respectively, they present a low prediction rate of ≈69% for the Flooding class. Based on our analysis (ET with Set 1 (subtype)), the Flooding class had a prediction rate of 89%, while LightGBM yielded a prediction rate of 93% for the Impersonation class. Again, due to the imbalanced nature of the wireless datasets, the F1 and AUC metrics should be the primary focus, and at least explicitly reported by the work.

As already pointed out, for comparison reasons, in Table 15 we provide results on three additional tests with four classes,

TABLE 15. Comparison with related work (all metrics are in percentages). A dash means "not provided," while N/A stands for "not applicable" and "Balanced" means "Balanced test set."

Model	Features	Classes	AUC	Prec	Recall	F1	Acc	Epochs	Balanced	k-fold	T.E. Time
AWID2 previous works											
LightGBM [5]	15	4	–	99.87	100	99.93	99.99	N/A	✓	✗	00:00:07
RF [9]	32/10/7/5*	–	100	100	100	100	99.99	N/A	✗	✓	00:02:58
DNN [11]	36	–	–	–	–	–	99.99	–	✗	✗	–
SVM [13]	20	4	–	–	99.92	99.94	99.97	N/A	✗	✗	02:59:49
Ensemble [12]	8	4	–	99.50	99.50	99.50	99.52	N/A	✗	✓	–
2-way WINDS [8]	19/7	4	–	–	–	–	99.42	N/A	✗	✗	–
Semi [10]	95	–	–	–	–	–	99.28	N/A	✗	✗	–
RF [7]	35	4	–	96	96	95	95.87	N/A	✗	✓	–
DNN+KNN [6]	154	2	–	86.15	92.18	89.06	94.81	100	✓	✗	–
This work. Best performers based on F1 score and number of classes for AWID2											
LightGBM	16	3	94.41	98.63	92.47	95.36	99.31	N/A	✗	✓	00:10:33
LightGBM	16	4	97.32	98.94	95.87	97.35	99.48	N/A	✗	✓	00:13:29
AWID3 previous works											
LightGBM [15]	5	3	–	82.33	82.57	82.18	82.57	N/A	✗	✗	–
This work. Best performer based on F1 score for AWID3											
ET	16	3	99.49	99.75	99.28	99.52	99.96	N/A	✗	✓	11:39:30

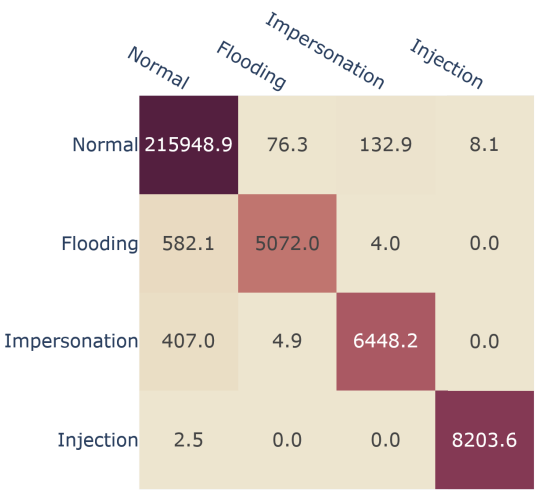


FIGURE 11. Confusion matrices average results across all folds for AWID2 model analysis with four classes.

namely, Normal, Flooding, Impersonation, and Injection. As illustrated in Figure 11, these experiments utilize ET with the reduced feature Set 1 (subtype), and LightGBM, over the 16-features set. As observed from Table 15 and Figure 11, in all these cases, the models presented a higher AUC score vis-à-vis the 3 class models. This is due to the Injection class, which presented a high F1 score of 97.35% for the LightGBM model; approximately 8,203 of the total 8,206 samples (99.96%) of the Injection class have been classified correctly. For ET the respective scores were 99.47% (missed only 42 samples) and 99.98% (missed just ≈1 sample). Overall, it can be said that this specific class improves the scores; this is expected as injection type of attacks are quite easily detectable.

Compared to the limited hitherto previous work on AWID3, the results given by the current study are superior. We argue that this is basically due to the nature of the features used in our experiments; note that the work in [15] also uses few features. An additional remark stemming from the study in [15] is that while the creation of sound (based on theory and a well-defined methodology) custom features can benefit the detection process, these features should be accompanied by a minimum set of row features, as detailed in sections III-A and IV-D.

F. TAKEAWAYS AND FUTURE DIRECTIONS

Based both on theory and empirical observations, this work provides a staple methodology for feature selection and data preprocessing targeting 802.11 IDS. Research should start with a minimum set of four features as explained in section IV-D, and depending on the case and the initial results, develop to a larger set of 16 or 19 features as detailed in sections 2 and IV-C, respectively.

The addition of custom features, tackling specific problems of the targeted IDS model, can also be a promising path forward. For instance, if the IDS model misplaces Flooding samples, a custom feature could be setting a counter on the Deauthentication or Disassociation frames; as a rule of thumb, a STA will never transmit more than 10 such frames within a short time window.

Another promising direction is unsupervised learning analysis as well as regression methods. By leveraging regression models along with proper feature selection as explained in sections III-A, IV-C, and IV-D may assist in creating other generalized ML models. And although the use of features such as the *wlan.ra* in regression is not prohibited, it is generally unsound to transform similar features to numerical. Simply put, these features should be handled as categorical and converted with the OHE technique. It is true that it is sometimes cumbersome to analyze these features due to the overhead they induce, that is, by adding multiple columns to the dataset and increasing the size and execution time

of the ML model. As already pointed out, researchers may experiment with custom features by combining MAC layer ones or otherwise.

Regarding AWID2 and DNN analysis, it was observed that the number of samples were rather inadequate to train the DNN models at a superior level. The DNN reached quickly, after 3 to 4 epochs, the optimal loss value, and after that it was struggling to escape this situation. As a result, the DNN model was trained for too little loss, namely, the loss was reduced by a negligible amount of 0.001 after every epoch. For this type of analysis, it is expected that more samples can produce more accurate results. Future research can examine the full AWID2 dataset, namely the *AWID-CLS-F-Trn* and *AWID-CLS-F-Tst* files, which when combined comprise ≈200M samples. Indicatively, a technique to handle this big number of samples is to split the dataset into smaller chunks of, say, 10M samples each, and analyze them separately. The extraction of an at least 15M samples part of the dataset, train only that, and use the remaining samples as a test set with a k-fold scheme is also a viable option. Last but not least, an additional option is to examine more advanced DNN models, such as time-series anomaly detection. Having two classes instead of three, and also examining the time relation between each sample could assist a DNN model in improving its prediction rate.

As it was expected, the analysis clearly showed that, in comparison to AWID2, the AWID3 dataset yields a superior prediction rate for legacy types of Flooding attacks. With a high degree of certainty, this result is due to the 802.11w amendment, also known as Protected Management Frames (PMF), which was always active during the generation of AWID3 [21]. Specifically, although not watertight, PMF is highly beneficial when dealing with Deauthentication and Disassociation attacks. PMF cryptographically protects sensitive management frames, namely Deauthentication, Disassociation, and Action frames. Therefore, for causing a DoS situation, the opponent can only transmit unprotected Deauthentication or Disassociation frames in an effort to disconnect a device [21] or rely on other types of frames such as the Simultaneous Authentication of Equals (SAE) ones [19]. Put simply, the features that make the difference here are the *frame.len*, which is increased for every protected frame, and the *wlan.fc.protected*, which is always set for protected Deauthentication or Disassociation frames.

Moreover, for AWID3, we noticed that DNN models confronted difficulties in detecting the attacks contained in *(Re)Assoc* CSV file, namely Association, Reassociation, and Beacon flooding. The frames involved in these attacks are transmitted unprotected and therefore may go unnoticed. The inclusion of the three additional features, as discussed in subsection IV-C, improved the average detection AUC score of DNN by 0.76%. This can be observed from the confusion matrices in Figures 7, 8, and 9. Generally, around 2K to 5K Flooding frames, which were misplaced as Normal traffic, belong to the (Re)Assoc CSV attack frames. To improve detection for this class, custom (engineered) features can be used, possibly along with Action frames. Recall that when an AP or STA receives an Association or Reassociation frame and the 802.11w (PMF) is enabled, it must transmit SA Queries (Action frames).

VII. CONCLUSION

In the era of Wi-Fi 6 implementing the IEEE 802.11ax standard, whilst Wi-Fi 7 (802.11be) is currently underway, the security traits of this type of technology have gradually evolved and ameliorated. Nevertheless, as the literature has repeatedly proven, 802.11 is not watertight in terms of security. This renders the need for IDS to promptly perceive and report ongoing attacks. The work at hand offers a solid, from both a theoretical and empirical viewpoint, process for selecting the right classification features and data preprocessing methods that possibly eliminate overfitting and result in more accurate, deployment-agnostic ML models. Exploiting the AWID family of benchmark datasets, we meticulously access the proposed process through an assortment of ML models, both shallow and deep learning. Moreover, it is demonstrated that an 802.11 IDS can be quite effective if just trained with a quartet of high importance features. Last but not least, we discuss shortcomings, misconceptions, or half-truths observed in the related work, suggesting the right direction.

In addition to the way forward mentioned in subsection VI-F, future work can assess the ability of the selected features to detect assaults manifested above the MAC layer, say, application layer volumetric ones. This direction is straightforward given that the AWID3 embraces several assaults manifested on the upper layers of the stack. The analysis of feature importance in, say, Linear models (*LinearSVC*), by calculating the coefficiency on each selected feature, is also a possible direction for subsequent work. Hyperparameter tuning techniques via the utilization of libraries like *Optuna*, *Hyperopt*, *Skikit-learn* (grid and random search), and *Scikit-Optimize* can be considered as well. The use of sampling techniques for the training set, including undersampling (time-series) and feature engineering, may also be beneficial in improving the detection ability of the algorithms or models depending on the particular case.

REFERENCES

[1] Kismet. *Kismet*. Accessed: Apr. 11, 2022. [Online]. Available: https://www.kismetwireless.net/

[2] AirSnort. *Airsnort*. Accessed: Apr. 11, 2022. [Online]. Available: https://ftp.unpad.ac.id/orari/library/library-sw-hw/linux-1/airsnort/AirSnort%20Homepage.htm

[3] Arubaos. *Arubaos*. Accessed: Apr. 11, 2022. [Online]. Available: https://www.arubanetworks.com/products/network-management-operations/arubaos/

[4] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st. Quart., 2016.

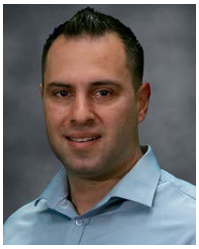
[5] S. Bhandari, A. K. Kukreja, A. Lazar, A. Sim, and K. Wu, "Feature selection improves tree-based classification for wireless intrusion detection," in *Proc. 3rd Int. Workshop Syst. Netw. Telemetry Anal.*, Jun. 2020, pp. 19–26.

[6] M. E. Aminanto and K. Kim, "Improving detection of Wi-Fi impersonation by fully unsupervised deep learning," in *Proc. Int. Workshop Inf. Secur. Appl.* Cham, Switzerland: Springer, 2017, pp. 212–223.

[7] F. D. Vaca and Q. Niyaz, "An ensemble learning based Wi-Fi network intrusion detection system (WNIDS)," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–5.
[8] A. A. Reyes, F. D. Vaca, G. A. C. Aguayo, Q. Niyaz, and V. Devabhaktuni, "A machine learning based two-stage Wi-Fi network intrusion detection system," *Electronics*, vol. 9, no. 10, p. 1689, Oct. 2020.
[9] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. Alessa, "Effective features selection and machine learning classifiers for improved wireless intrusion detection," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Jun. 2018, pp. 1–6.
[10] J. Ran, Y. Ji, and B. Tang, "A semi-supervised learning approach to IEEE 802.11 network anomaly detection," in *Proc. IEEE 89th Veh. Technol. Conf.*, Apr. 2019, pp. 1–5.
[11] S. Rezvy, Y. Luo, M. Petridis, A. Lasebae, and T. Zebin, "An efficient deep learning model for intrusion classification and prediction in 5G and IoT networks," in *Proc. 53rd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2019, pp. 1–6.
[12] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107247.
[13] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.
[14] A. Lazar, A. Sim, and K. Wu, "GPU-based classification for wireless intrusion detection," in *Proc. Syst. Netw. Telemetry Anal.*, Jun. 2020, pp. 27–31.
[15] A. Agrawal, U. Chatterjee, and R. R. Maiti, "CheckShake: Passively detecting anomaly in Wi-Fi security handshake using gradient boosting based ensemble learning," Cryptol. ePrint Arch., Tech. Rep. 2021/1702, 2021. [Online]. Available: https://ia.cr/2021/1702
[16] S. M. Kasongo and Y. Sun, "A deep learning method with filter based feature engineering for wireless intrusion detection system," *IEEE Access*, vol. 7, pp. 38597–38607, 2019.
[17] S. M. Kasongo and Y. Sun, "A deep long short-term memory based classifier for wireless intrusion detection system," *ICT Exp.*, vol. 6, no. 2, pp. 98–103, Jun. 2020.
[18] S. M. Kasongo and Y. Sun, "A deep gated recurrent unit based model for wireless intrusion detection system," *ICT Exp.*, vol. 7, no. 1, pp. 81–87, Mar. 2021.
[19] E. Chatzoglou, G. Kambourakis, and C. Kolias, "How is your Wi-Fi connection today? DoS attacks on WPA3-SAE," *J. Inf. Secur. Appl.*, vol. 64, Feb. 2022, Art. no. 103058.
[20] O. Kanhere and T. S. Rappaport, "Position locationing for millimeter wave systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 206–212.
[21] E. Chatzoglou, G. Kambourakis, and C. Kolias, "Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset," *IEEE Access*, vol. 9, pp. 34188–34205, 2021.
[22] D. Schepers, A. Ranganathan, and M. Vanhoef, "On the robustness of Wi-Fi deauthentication countermeasures," in *Proc. 15th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, New York, NY, USA, May 2022, pp. 245–256, doi: 10.1145/3507657.3528548.

EFSTRATIOS CHATZOGLOU received the M.Sc. degree in security of information and communication systems from the University of Aegean, Samos, Greece. He was a Web Developer of integrated health information system web application with the Hellenic Army General Stuff, Greece. He is currently a Penetration Tester with the Hellenic National Defense General Stuff, Greece, and a Research Associate at the Info-Sec-Laboratory, University of Aegean. His research interests include the fields of wireless and cellular networks security, the IoT networks security, android application security, web application security, and machine learning.

GEORGIOS KAMBOURAKIS is a Full Professor with the Department of Information and Communication Systems Engineering, University of the Aegean, Greece. He was the Head of the Department from September 2019 to October 2019, and was the Director of Info-Sec-Laboratory from September 2014 to December 2018. Currently, he is on unpaid leave from the University, while he is working for the European Commission at the European Joint Research Centre (JRC), Ispra, Italy. He has more than 150 refereed publications in his research areas. His research interests include the fields of mobile and wireless networks security and privacy, VoIP security, IoT security and privacy, DNS security, and security education. More info at: http://www.icsd.aegean.gr/gkamb



CONSTANTINOS KOLIAS received the doctorate degree from the University of the Aegean, in 2014. He was a Research Assistant Professor at the CS Department, George Mason University. He joined the Computer Science Department at the University of Idaho, in 2018. He is also active in the design of intelligent IDS with a special interest in privacy preserving distributed IDS. His main research interests include security and privacy for the IoT and critical infrastructures, mobile and wireless communications security, and privacy enchasing techniques for the internet. More info at: https://www.uidaho.edu/engr/departments/cs/our-people/faculty/constantinos-kolias

CHRISTOS SMILIOTOPOULOS received the B.Sc. degree (Hons.) in computing and IT from the Open University, U.K., in 2020, and the M.Sc. degree in security of information and communication systems from the University of the Aegean, in 2022, where he is pursuing the Ph.D. degree. He is currently a UAV SPERWER Mission Planner and a CIS Officer with the Hellenic Army. His research interests include the fields of advanced machine learning techniques in scenarios of mobile and end-point forensics, android application security, wireless network security, android application development, and java and python programming.