Wi-Fi Network Intrusion Detection: Enhanced with Feature Extraction and Machine Learning Algorithms

1st Şevval Şolpan
Software Engineering
Kocaeli University
Kocaeli, Turkiye
sevval.solpan@kocaeli.edu.tr

2nd Hakan Gündüz
Software Engineering
Kocaeli University
Kocaeli, Turkiye
hakan.gunduz@kocaeli.edu.tr

3rd Kerem Küçük
Software Engineering
Kocaeli University
Kocaeli, Turkiye
kkucuk@kocaeli.edu.tr

Abstract*—Wireless Fidelity (Wi-Fi) is one of the most popular technologies for Internet connection. It helps users to connect to other devices by simply typing the correct password. It contributes to a great extent to building environments powered by IoT in schools, offices, homes, and cities. However, it has vulnerabilities, causing others to suffer low quality service or personal information theft when misused. Misusing is possible in several ways, and it is often caused by a network attack. To maintain the security of a network, network intrusion detection systems are important. For this reason, this study focused on network intrusion detection and used publicly available Aegean Wi-Fi Intrusion Dataset 3 (AWID3). We have used K-Nearest Neighbors, Support Vector Machine, and Gradient Boosting Trees algorithms for attack detection. In addition, we used time-series data and extracted features using two methods: Convolutional Neural Network (CNN) and statistical measures, to observe the effect on learning and facilitate the delivery of time-series data to machine learning algorithms. To the best of our knowledge, we are the first to conduct experiments on the AWID3 dataset using a combination of machine learning algorithms, feature extraction methods, and time-series data for network intrusion detection. We conducted six experiments in total, and the results of our best-performed experiments are as follows: accuracy: 0.978, precision: 0.978, recall: 0.978, F1 score: 0.978.

Keywords — Wi-Fi, IEEE 802.11, intrusion detection, attack classification, machine learning.

I. INTRODUCTION

Wireless Fidelity (IEEE 802.11, Wi-Fi) plays a crucial role in the Internet of Things (IoT) because it eases the integration of devices into IoT systems and operates well. For instance, although IoT devices run with low energy consumption, Wi-Fi technology efficiently supports high-speed data transfer. The wireless Internet connection provides comfort, flexibility, and portability. Wi-Fi networks are used in many environments, such as homes, office buildings, and public places. Typing the correct password is sufficient to create smart homes or environments. However, Wi-Fi networks have vulnerabilities arising from their structure. Misusing them may cause damage to others.

Public Wi-Fi networks are the most dangerous networks because they are used by many. Once an unauthorized user has access to a public network, that user can attack other users in the same network or outside of the network, e.g., by sending excessive network packets to other users [1, 2]. These attacks may cause users to lose network connections, connect to rogue access points, spend longer connection times, install malware on devices, and route untrue websites. Thus, all attacks result in unauthorized Access to and theft of private information. The increased variety of attacks and incidents requires the detection of network attacks. Note that checking the existence of any attack in the network, regardless of the type, is the target problem in this study.

The detection of network attacks requires analysis of network traffic. Accordingly, this study used the publicly available Aegean Wi-Fi Intrusion Dataset 3 (AWID3), which contains data related to 13 network attacks. The dataset contains 254 attributes. It contains over 30 million normal traffic data and over 6 million malicious traffic data [3]. Because these models are easier to comprehend, we employed machine learning algorithms for network traffic data analysis. Network intrusion detection is considered a classification problem in machine learning. It has been observed that many studies have utilized machine learning algorithms: Logistic Regression, Decision Tree (DT), Random Forest (RF), Reduced Error Pruning Tree (REP-T), J48, Hoeffding Tree (HT), Half Space Tree (HST), Accuracy Weighted Ensemble Classifier (AWEC), and Decision Stump (DS) [4, 5, 6]. The algorithms used in this study were K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gradient Boosting Trees (GBT).

Furthermore, there is a connection between some network traffic events as network attacks might take time to complete, which we explained in detail in the methodology section. We wanted to use this meaningful temporal relationship and included a data preprocessing phase that produced 100-row time-series data fit for processing. At this stage of our study, the time-series data input to machine learning algorithms becomes challenging. As a solution to this, we used two feature extraction methods to ease the delivery of time-series data to machine learning algorithms, another topic we explained in detail in the methodology section. Previous studies have used various methods, such as autoencoder, neural network, ANOVA F-Test, Mutual Information, Recursive Feature Elimination, Permutation Importance, Convolutional Neural Network (CNN), and statistical measures [7, 8, 9, 10, 11, 12]. To the bset of our knowledge, we are the first to conduct experiments on the AWID3 dataset by adding a separate feature extraction phase to the classification of network intrusion detection. Contributions of this paper are as follows:

- To shape the dataset into 100-row time-series data
- Apply two feature extraction methods: CNN and statistical measures
- To conduct six experiments using a combination of feature extraction methods and machine learning algorithms

979-8-3315-3149-2/24/$31.00 ©2024 IEEE

- To results of our study and their comparison with those of other studies

This paper is organized as follows: Detailed information about the dataset and machine learning algorithms is presented in the second section. The data preprocessing, feature extraction, and classification phases are explained in the third section. The fourth section presents the findings of this study and compares its results with those of other studies.

II. BACKGROUND

This section provides detailed information about the dataset and the machine learning algorithms used in this study.

A. *Dataset*

The AWID3 is a publicly available labeled dataset and created by adding modern network attacks to the AWID to detect network intrusions. Physical layer attacks are irrelevant to the AWID3 dataset [3, 15]. AWID3 categorizes these attacks into 4.

The first category includes the attacks related only to IEEE 802.11. Here, deauthentication, disassociation, reassociation, rogue AP, krack, and kr00k are considered. The second category includes evil-intentioned attacks directed at users in the same local area network. Attacks in the second category are SSH brute force, botnets, and malware. The third category includes attacks that start in a local area network and are hostile to targets outside the local area network. SQL injection and SSDP amplification are attacks in this category. Some attacks misuse multiple protocol vulnerabilities. Thus, they can generate multilayer attacks. The fourth category includes multilayer attacks called “evil twin” and website spoofing.

AWID3 contains data in 13 folders, each related to a different attack. The folders are comprise varying numbers of CSV files comprising 254 attributes. The dataset contains over 30 million normal traffic data and over 6 million malicious traffic data.

B. *Machine Learning Algorithms*

We evaluated the performances of three machine learning algorithms on the AWID3: KNN, SVM, and GBT. We selected these algorithms because we wanted to observe the performance of the algorithms with different complexities. The KNN and SVM algorithms are both supervised learning algorithms. However, the KNN algorithm is simple and requires minimal parameter tuning. On the other hand, SVM is effective for high-dimensional data and provides flexibility for tuning with different kernels. GBT is an ensemble learning method. It builds new models, all of which attempt to correct errors in the previous models. These new models are usually decision trees, and classification using a GBT is time-consuming.

III. METHODOLOGY

Our methodology involves data preprocessing, feature extraction, and classification. We obtained data blocks, which we used as input for feature extraction at the end of the preprocessing phase. In the feature extraction phase, we utilized two different methods to observe the effect of each extraction method on classification success. In addition, we employed three machine learning algorithms to increase the number of solutions to network intrusion detection problems. The general structure is illustrated in Figure 1.

A. *Data Preprocessing*

We mentioned earlier that AWID3 contains 13 folders of data. This study utilizes the data from each folder. Each data file consisted of 254 attributes. We used the attributes that Chatzoglou et al. and Ahmad et al. did [16]. However, we have added an additional attribute, frame.time_epoch, to be removed in the further stages of data preprocessing. We have listed the attributes used in our study in Table I.

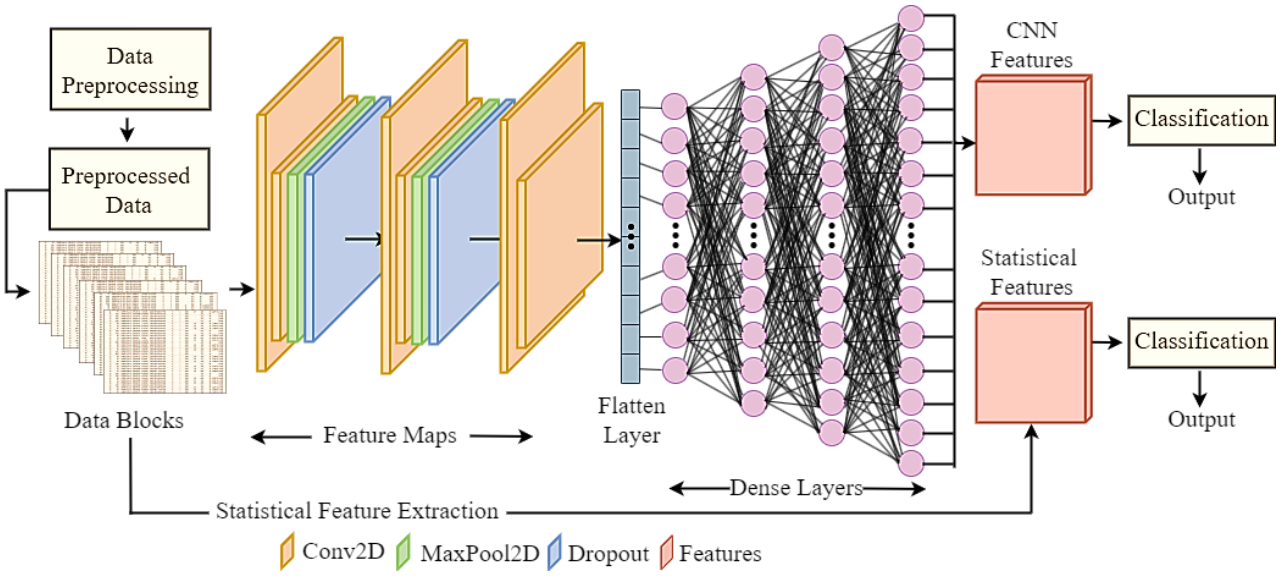

Figure 1. Graphical representation of our methodology

Starting from the first attack folder, deauthentication, we created a data frame consisting of data belonging to the attributes mentioned earlier from all CSV files in the same folder. Second, we removed all rows containing NaN values from the data frame. Third, we balanced the data to avoid the effects of unbalanced data on learning using a method similar to that used by Ahmad et al. followed [17]. We randomly took some data from each attack folder to protect the normal traffic to malicious traffic data ratio, which Ahmad et al. preferred to use as 2:1 in their study. Table I lists the amount of data used in this study according to attack type, normal traffic, and malicious traffic. In the fourth step, we sorted the data according to frame.time_epoch after randomly selecting normal and traffic data.

TABLE I. TABLE I. FEATURES USED IN THE STUDY

Feature List	
frame.len	wlan.fc.type
frame.time_epoch	wlan.fc.subtype
radiotap.length	wlan.fc.ds
radiotap.dbm antsignal	wlan.fc.frag
wlan.duration	wlan.fc.retry
radiotap.present.tsft	wlan.fc.pwrmgt
radiotap.channel.freq	wlan.fc.moredata
radiotap.channel.flags.cck	wlan.fc.protected
radiotap.channel.flags.ofdm	Label

Machine learning algorithms work with numerical data. However, some attributes contain string values, e.g., radiotap.dbm_antsignal, radiotap.present.tsft, wlan.fc.ds, wlan.fc.subtype, and label. Thus, we checked the attributes using the AWID3 dataset because some values do not exist in all CSV files. In the fifth step, we used the unique values of the attributes to create a list, and we applied integer mapping according to the list. In the sixth step, we applied min-max normalization to the columns except for frame.time_epoch and label. We removed the "frame.time_epoch" column from the dataset after the sixth step because it was no longer needed.

Typically, network intrusion attacks do not occur instantly. They take time to complete. Thus, the temporal relationship gains importance relative to network intrusion detection. Based on this, we assumed a window with 100 rows of length and the same column number as the data. We have slid this imaginary window over 50-row steps at a time and obtained data blocks, which occur 100 rows of temporally ordered data without losing the temporal relationship.

Network intrusion attacks can occur in a sequence of network traffic events. Although a network event is harmless, it may be part of an attack or may prepare the ground for an attack. For example, deauthentication attacks are frequently used to prepare the ground for more complex attacks [3]. Thus, other network events become suspicious even if there is a single attack in the session. Based on this, we checked each data block in the final data preprocessing step. If the data block does not contain any row labeled as an attack, all rows of data are relabeled as 0. In other states of the data blocks, all rows were relabeled as 1.

All steps were applied to data in the other attack folders. (In the SSDP folder, we applied the preprocessing steps only for the first 75 CSV files different from others due to insufficient memory.) After preprocessing, we obtained data blocks and combined them according to the attack folder order in the further stages of the study. Table II lists the block quantities obtained from the AWID3 dataset.

TABLE II. TABLE II. QUANTITIES OF NORMAL TRAFFIC, MALICIOUS TRAFFIC DATA AND DATA BLOCKS

Attacks	**Quantities**		
	Normal Traffic	***Malicious Traffic***	***Blocks***
Deauthentication	76000	38000	2279
Disassociation	14000	7000	419
Reassociation	10000	5000	299
Rogue AP	2000	1000	59
Krack	98000	49000	2939
Kr00k	300000	150000	8999
SSH Brute Force	20000	1000	599
Botnet	100000	50000	2999
Malware	200000	100000	5999
SQL Injection	4000	2000	119
SSDP Amplification	200000	100000	5999
Evil Twin	200000	100000	5999
Website Spoofing	200000	100000	5999
Total	1406000	703000	42707

B. Feature Extraction

We have turned the data into data blocks to protect temporal relationships and have selected them to use machine learning algorithms for classification. When considering of the structure of a data block, the structure is a 2D shape similar to an image. Additionally, the values in a data block and the pixel values in an image are similar. However, we could not provide 2D structured data to the machine learning algorithms as input. Therefore, we extracted the features of the attributes. We utilized two methodologies to observe the effect of feature extraction methodologies on classification success: CNN and statistical feature extraction.

1) Statistical Feature Extraction

We employed statistical feature extraction because statistical data analysis is one of the preferred ways to understand data. Extraction is realized by calculating the minimum, maximum, mean, variance, first quartile, second quartile, and third quartile of each attribute in a data block. After extracting the statistical features of a data block, we obtained a single row of feature data with 112 columns. At the end of the extraction, we obtained equal rows of feature data relative to the number of data blocks.

2) Feature Extraction with CNN

We utilized a 2D-CNN model because it is a deep learning model that performs well at feature extraction and is adaptable to 1D or 2D structured data. The layers of the CNN model we used in this study were convolution with 16-filter, convolution with 8-filter, max pooling, dropout, convolution with 16-filter, convolution with 8-filter, max pooling, dropout, convolution with 16-filter, convolution with 8-filter, flatten, dense, dense, dense, and dense as depicted in Figure 1.

The kernel size is 3x3, the activation function is ReLU, and the padding was "same" in all convolution layers except filter numbers. The filter numbers were 16 or 8. The 16-filter convolution layers always came before the 8-filter convolution layers in the model. Multiple convolution layers were employed to extract complex features by increasing the depth of the model. In addition, we wanted to prevent parameter increases and determined the filter numbers from 16 to 8 in the convolutional layers. We added max pooling layers with a 2x2 pool size and a default striding value to the model to eliminate redundant features. We applied 10% dropouts to avoid overfitting. After the extraction layers, the extracted features are given to the flatten layer, which transforms the 2D structured data into a 1D vector. The 1D data are processed in the 4-hidden-layered fully connected network, whose neurons gradually increase. The numbers of neuron was 16, 32, 64, and 112. Then, the features are input to the machine learning algorithms. We ended the model with 112 neurons in the last hidden layer because the statistical features would be 112 columns of data. We made the features equal in terms of column and row numbers for comparison.

We added layers and configured the model parameters in trial cases. The model and its parameters were the same in all experiments. We added layers to the model using TensorFlow's Keras library. While training the 2D-CNN, we used the Adam optimizer, whose learning rate was 0.0001, from TensorFlow's Keras library. We split the data to ensure that 30% of all features were test data and 70% were training data using the Scikit-learn library. The parameter called random_state was 42 while splitting. The number of epochs is 10, and the batch size was 64. We also applied early stopping, whose parameters are monitor=val_loss, patience=1, and restore_best_weights=True.

C. Classification

As they perform well in classification tasks, we used three machine learning algorithms in the classification phase: KNN, SVM, and GBT (KNeighborsClassifier, SVC, and GradientBoostingClassifier from the Scikit-learn library). We have already used two feature extraction methods. Thus, we conducted six experiments in total. At the beginning of the classification, we split our features to obtain 20% of all features as test data. 80% of all features were training data. The parameter called random_state was 42 while splitting. Then, we conducted training and testing using the machine learning models. We applied K-Fold Cross-Validation from the Scikit-learn library to obtain more trusted results using the same parameters during all classification processes. They are n_splits=10, random_state=42, and shuffle=True.

In addition, we repeated our experiments using various parameters depending on the classification algorithm. For example, the classification success of KNN algorithms depends on the parameter value n_neighbors. Therefore, we repeated the experiments for classifying with KNN by changing the 'n_neighbors' parameter value. These values are 1, 3, 5, 7, 9, 11, 13, and 15. In the classification experiments using SVM, we repeated the experiments by changing the kernel, gamma, and C parameter values. Kernel values are 'linear', 'rbf', 'poly', and 'sigmoid'. Gamma values are "scale" and "auto". The C values were 0.1, 1, 10, and 100. In the GBT classification experiments, we repeated the experiments by changing the learning_rate, n_estimators, and max_depth parameter values. The learning_rate values are 0.001, 0.01, and 0.1. The 'n_estimators' values are 100, 200, and 500. The max_depth values are 3, 4, and 8.

IV. RESULTS

Although we conducted many experiments, we presented the results of the experiments when the parameters gave the best results. This section presents the loss and accuracy curves of the feature extractions with 2D-CNN, the confusion matrixes, and the results of six experiments.

A. Loss and Accuracy Curves of Feature Extraction

Figure 2 shows the training and validation accuracies of feature extraction in the experiment in which classification was realized with GBT. Figure 3 shows the training and validation losses of the feature extraction in the experiment in which classification was realized with GBT.

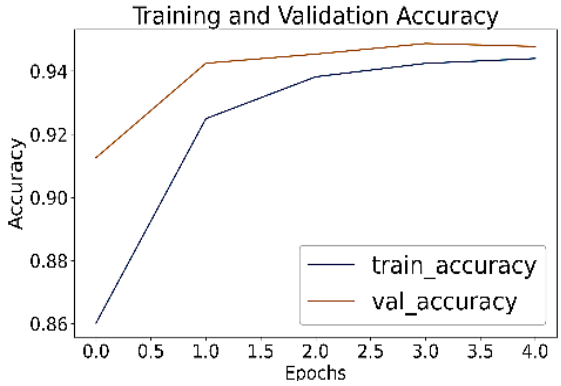

Figure 2. Training and validation accuracy of the feature extraction with 2D-CNN (Classification: GBT)

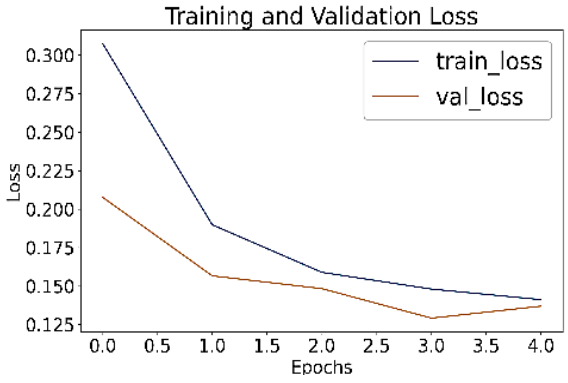

Figure 3. Training and validation losses of the feature extraction with 2D-CNN (Classification: GBT)

After the third epoch, val_accuracy decreased slightly, and val_loss increased. After the third epoch, the model starts overfitting. However, the model was stopped early to prevent overfitting and the best values were restored during training.

B. Confusion Matrix of The Best Performed Experiment

The vertical axis in the matrix represents true values. The horizontal axis in the matrix represents the predicted values. The 0 on the axis represents the normal traffic class, and the 1 on the axis represents the malicious traffic class or the existence of an attack. The value at the intersection of 0 of the true class and 0 of the predicted class indicates the amount of normal traffic data predicted as normal traffic data. The value at the intersection of 1 in the true class and 0 in the predicted class indicates the amount of malicious traffic data predicted as normal traffic data. The value at the intersection of 0 in the true class and 1 in the predicted class indicates the amount of normal traffic data predicted as malicious traffic data. The value in the intersection of 1 of the true class and 1 of the predicted class shows the amount of malicious traffic data predicted as malicious traffic data.

The experiment, whose features were extracted by statistical measures and classified by GBT, performed best and confusion matrix of the best performing experiment is illustrated in Figure 4. The amount of normal traffic data predicted as normal traffic data is 3504. The amount of malicious traffic data predicted as normal traffic data is 129. The amount of normal traffic data predicted as malicious traffic data is 56. The amount of malicious traffic data predicted as malicious traffic data is 4853.

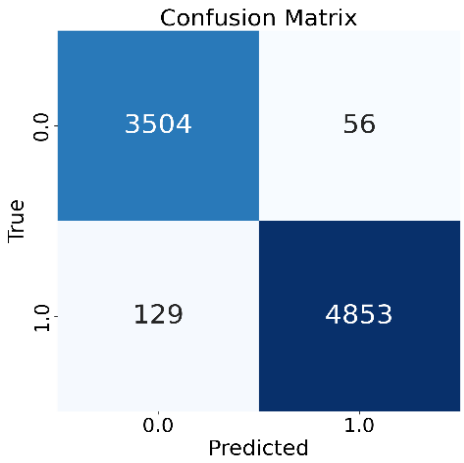

Figure 4. The best performing experiment (Feature extraction: Statistical measures, Classification: GBT)

C. The Results of The Six Experiments and Their Comparison with Other Studies

Table III presents the results of the six experiments conducted in this study. Accuracy represents the quantity of samples that the model predicted correctly. The results demonstrate that our experiments were successful, with the lowest accuracy of 0.945. However, GBT demonstrated the best performance with an accuracy of 0.978. Precision represents the quantity of the predicted positive predictions by the model. The results demonstrate that the lowest precision was 0.946, and the highest precision was 0.978.

TABLE III. TABLE III. THE RESULTS OF SIX EXPERIMENTS

No	Experiments		Metrics				K-Fold Cross-Validation	
	Feature Extraction	*Classifiers*	*Accuracy*	*Precision*	*Recall*	*F1-Score*	*Accuracy*	*Standard Deviation*
1	CNN	KNN	0.946	0.947	0.946	0.946	0.945	0.005
2		SVM	0.945	0.946	0.945	0.945	0.950	0.004
3		GBT	0.950	0.951	0.950	0.950	0.951	0.004
4	Statistical Measures	KNN	0.948	0.948	0.948	0.948	0.946	0.002
5		SVM	0.965	0.966	0.965	0.966	0.965	0.003
6		**GBT**	**0.978**	**0.978**	**0.978**	**0.978**	**0.979**	**0.002**

The recall represents the number of positive predictions that the model truly predicted as positive. The recall values of the worst-performed experiment is 0.945, whereas the best-performed experiment is 0.978. The F1 score is the harmonic mean of precision and recall. It evaluates the model's performance, whether balanced or not, according to precision and recall. The worst value was 0.945, while the best value was 0.978. According to accuracy, precision, recall, and F1 score, experiment 2 gave the lowest values. The best performing experiment was experiment 6, which produced the highest values.

The accuracy obtained from K-fold cross-validation provides information about the general performance of the model on different datasets. The standard deviation indicates the change in the accuracy of the model depending on the number of folds. If the standard deviation value is low, then the change is also low. However, if the standard deviation value is high, then the change is high. Experiment 1 had the lowest accuracy obtained by K-fold cross validation (0.945) and the highest standard deviation, 0.005. Accuracy of 0.945, which is the lowest among the experiments, is a high value, indicating that the model performs well. A 0.005 standard deviation indicates that the model's performance was not steady. Experiment 6 outperforms the others in both accuracy and standard deviation obtained from K-fold cross-validation. It gives the highest accuracy and the lowest standard deviation, 0.979 and 0.002, respectively, which means that experiment 6 is the best performed and most stable of the six.

Although it has one of the lowest accuracies obtained from K-fold cross-validation, another experiment is as stable as experiment 6, which is experiment 4.

If we compare the experiments according to the feature extraction methods, the experiments using statistical measures perform better than the experiments utilizing a CNN. If we compare the experiments according to machine learning algorithms, the experiments are sorted as follows: SVM, KNN, and GBT in order of increasing metric performance in the experiments using CNN. The experiments were sorted as KNN, SVM, and GBT in order of metric performance in the experiments using statistical measures.

Table IV compares the results of our and other studies. According to accuracy, if we sort models in decreasing order, the order is RF, REP-T, J48, RT, DT, RF, HT, HT, SM-GBT, SM-SVM, LightGBM, CNN-GBT, SM-KNN, DS, CNN-KNN, CNN-SVM, XGBoost, AWEC, KNN, and HST [5, 6, 13, 14]. In order, tree classifiers perform better than most. Our best performed model also employs the gradient boosting tree classifier.

One of the previous study indicated that feature extraction has been overlooked in the literature [18]. A previous study used KNN as we did, and our methodology surpasses theirs, which proves that combining KNN for classification with statistical measures for feature extraction is effective. The enhancement of feature extraction methods on classification accuracy was also discussed and supported by experiments in another previous study, which focused on the effects of feature extraction methods on network intrusion detection success.

According to the other metrics, we can compare our results to only LightGBM and XGBoost due to missing information [14]. In terms of precision, Precision represents the accuracy of the model's positive predictions. If we sort algorithms in decreasing order, the order is LightGBM, XGBoost, SM-GBT, SM-SVM, CNN-GBT, SM-KNN, CNN-KNN, and CNN-SVM. The LightGBM and XGBoost models outperformed our models according to precision metrics, with values of 0.997 and 0.986, respectively, while our best precision value was 0.978. However, the LightGBM and XGBoost models' accuracies are lower than their precision and some of our models'. This means that the LightGBM and XGBoost models may miss some positives when the accuracy is lower than the precision. The LightGBM and XGBoost models are more selective and careful in predicting positive values, whereas they are not the same in predicting negative values. On the other hand, our models have precision and accuracy values with very little difference, which means they perform almost the same in predicting positive and negative values.

Regarding recall, if we sort models in decreasing order, the order is SM-GBT, SM-SVM, LightGBM, CNN-GBT, SM-KNN, CNN-KNN, CNN-SVM, and XGBoost. The LightGBM's was 0.958, and our two models, SM-GBT and SM-SVM, were better, with values of 0.978 and 0.965, respectively. Due to recall representing the number of positive predictions that the model truly predicted as positive, lower recall values compared to the precision values of the models also indicate that the LightGBM and XGBoost models may miss some positives. Our models have the same or similar values; thus, our models perform almost the same.

Regarding the F1 score, if we sort models in decreasing order, the order is SM-GBT, SM-SVM, CNN-GBT, SM-KNN, LightGBM, CNN-KNN, CNN-SVM, and XGBoost. The F1 score was the same. The SM-GBT and SM-SVM models produced better results, yielding 0.978 and 0.966, respectively, while the LightGBM model produced 0.947. The F1 score evaluates the performance of a model regarding balance. Higher F1 score values obtained using SM-GBT and SM-SVM demonstrate that our models correctly predicted the positive and negative values better than the LightGBM and XGBoost models.

The results obtained for the evaluation metrics demonstrate that although the proposed models are not the best in every metric, they are effective and competent. Our best-performed model obtained 0.978 accuracy, which is a high value, and balanced performance in predicting negative and positive values.

TABLE IV. TABLE IV. COMPARISON WITH OTHER STUDIES

Models	**Evaluation Metrics**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
RF [4]	0.999	-	-	-
REP-T [4]	0.999	-	-	-
J48 [4]	0.999	-	-	-
HT [4]	0.997	-	-	-
RT [4]	0.999	-	-	-
DS [4]	0.948	-	-	-
HT [5]	0.98	-	-	-
KNN [5]	0.88	-	-	-
AWEC [5]	0.92	-	-	-
HST [5]	0.88	-	-	-
DT [6]	0.999	-	-	-
RF [6]	0.999	-	-	-
LightGBM [7]	0.957	0.997	0.958	0.947
XGBoost [7]	0.941	0.986	0.906	0.915
CNN-KNN	0.946	0.947	0.946	0.946
CNN-SVM	0.945	0.946	0.945	0.945
CNN-GBT	0.950	0.951	0.950	0.950
SM-KNN	0.948	0.948	0.948	0.948
SM-SVM	0.965	0.966	0.965	0.966
SM-GBT	0.978	0.978	0.978	0.978

V. CONCLUSION

This comparison shows that our results are suitable and can be improved. Our methodology is effective because feature extraction enhances the detection success of machine learning algorithms. However, delivering time-series data to machine learning algorithms was a challenge during the experiments. In future, we plan to conduct experiments with more complex classification methods, e.g., deep learning methods. One of the reasons we plan to use deep learning methods is that delivering time series data will be easier because some deep learning methods process sequential data without extra processes. The feature extraction methods positively affected the classification success, as demonstrated

by the experiments. Thus, we investigated the effect of time-series data of different sizes on classification. Because AWID3 contains network traffic data related to 13 attacks, multiclass classification is also possible in future work.

VI. REFERENCES

[1] Natkaniec, M., and Bednarz, M., Wireless local area networks threat detection using 1D-CNN, *Sensors 23.12 (2023): 5507, 2023.*

[2] Farea, A. H., Küçük, K. "Machine Learning-based Intrusion Detection Technique for IoT: Simulation with Cooja", *International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.1, pp.1-23, 2024.* DOI:10.5815/ijcnis.2024.01.01

[3] 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,* doi: 10.1109/ACCESS.2021.3061609.

[4] Škrak, P. Lehoczký, P., Bencel, R., Galinski, M. and Kotuliak, I., "Improved Preprocessing for Machine Learning Intrusion Detection in IEEE 802.11,"*2022 14th IFIP Wireless and Mobile Networking Conference (WMNC),* Sousse, Tunisia, 2022, pp. 118-122, doi: 10.23919/WMNC56391.2022.9954288.

[5] Kamble, A. and Kshirsagar, D., "Feature Selection in Wireless Intrusion Detection System for Evil Twin Attack Detection,"*2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT),* Dehradun, India, 2023, pp. 1-5, doi: 10.1109/CISCT57197.2023.10351382.

[6] A. Torrres, “Wifi anomaly behavior analysis based intrusion detection using online learning.” *International Foundation for Telemetering,* 2021.

[7] J. Lee, J. Pak and M. Lee, "Network Intrusion Detection System using Feature Extraction based on Deep Sparse Autoencoder," *2020 International Conference on Information and Communication Technology Convergence (ICTC),* Jeju, Korea (South), 2020, pp. 1282-1287, doi: 10.1109/ICTC49870.2020.9289253.

[8] A. Wang, X. Gong and J. Lu, "Deep Feature Extraction in Intrusion Detection System," *2019 IEEE International Conference on Smart Cloud (SmartCloud),* Tokyo, Japan, 2019, pp. 104-109, doi: 10.1109/SmartCloud.2019.00028.

[9] J. Liu and S. S. Chung, "Automatic Feature Extraction and Selection For Machine Learning Based Intrusion Detection," *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI),* Leicester, UK, 2019, pp. 1400-1405, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00254.

[10] K. Janaki, D. V. Ravi Kumar, C. V. Murali krishna, A. Sravani and G. J. Moses, "Enhancing Intrusion Detection with Advanced Feature Extraction Through Machine Learning and Deep Learning Methods," *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS),* Gurugram, India, 2024, pp. 1-6, doi: 10.1109/ISCS61804.2024.10581050.

[11] C. Chen, X. Xu, G. Wang and L. Yang, "Network intrusion detection model based on neural network feature extraction and PSO-SVM," *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP),* Xi'an, China, 2022, pp. 1462-1465, doi: 10.1109/ICSP54964.2022.9778404.

[12] V. A and A. Vikas Singh, "Analysis of Traffic Sampling on Machine Learning Based Network Intrusion Detection," *2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon),* Singapore, Singapore, 2023, pp. 415-420, doi: 10.1109/SmartTechCon57526.2023.10391803.

[13] Saini, R., Halder, D. and Baswade, A. M., "RIDS: Real-time Intrusion Detection System for WPA3 enabled Enterprise Networks," *GLOBECOM 2022 - 2022 IEEE Global Communications Conference,* Rio de Janeiro, Brazil, 2022, pp. 43-48, doi: 10.1109/GLOBECOM48099.2022.10001501.

[14] Bhutta, A. A., Nisa, M. U., and Mian, A. N., Lightweight real-time WiFi-based intrusion detection system using LightGBM. *Wireless Networks, 30(2), 749-761, 2024.*

[15] Kolias, C., Kambourakis, G., Stavrou, A. and Gritzalis, S., "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," *IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 184-208, Firstquarter 2016,* doi: 10.1109/COMST.2015.2402161.

[16] Chatzoglou, E., Kambourakis, G., Kolias, C. and Smiliotopoulos, C., "Pick Quality Over Quantity: Expert Feature Selection and Data Preprocessing for 802.11 Intrusion Detection Systems," *IEEE Access, vol. 10, pp. 64761-64784, 2022,* doi: 10.1109/ACCESS.2022.3183597.

[17] Ahmad, R. S., Ali, A. H., Kazim, S. M. and Niyaz, Q., "A GAF and CNN based Wi-Fi Network Intrusion Detection System," *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* Hoboken, NJ, USA, 2023, pp. 1-6, doi: 10.1109/INFOCOMWKSHPS57453.2023.10226036.

[18] T. -C. Vuong, H. Tran, M. X. Trang, V. -D. Ngo and T. V. Luong, "A Comparison of Feature Selection and Feature Extraction in Network Intrusion Detection Systems," *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC),* Chiang Mai, Thailand, 2022, pp. 1798-1804, doi: 10.23919/APSIPAASC55919.2022.9979923.