

Article

Enhancing Wi-Fi Network Security Through Stacked Machine Learning Models

Filip Winiarski and Marek Natkaniec * 

Institute of Telecommunications and Cybersecurity, AGH University of Krakow, al. Mickiewicza 30, 30-059 Krakow, Poland; fwini@student.agh.edu.pl

* Correspondence: natkanie@agh.edu.pl

Abstract

The paper presents a wireless intrusion detection system (WIDS) based on model stacking for IEEE 802.11 Wireless Local Area Networks (WLANs). The proposed system utilizes model stacking, a machine learning (ML) technique in which the predictions of several ML models, called base classifiers, are combined to improve the overall performance of the system. Various stacking algorithms and ML architectures, including recurrent neural networks (RNN) and Convolutional Neural Networks (CNN), are implemented and compared. For the training of ML models and the evaluation of the proposed solution, the Aegean Wi-Fi Intrusion Dataset 3 (AWID3) was used. Moreover, the feature selection and data preprocessing are described in detail. The model stacking algorithms were shown to improve the WIDS performance in multiple scenarios, with the best ones achieving very high 99.878% accuracy and 99.507% F1 score for binary and 99.858% accuracy and 99.616% F1 score for multiclass classification. The experimental results demonstrate that selected stacking configurations achieve improved detection performance compared with the individual base classifiers while highlighting the influence of the chosen aggregation strategy. Most notably, the implementation of model stacking is shown to greatly reduce the number of false positives in most of the tested scenarios, which is a pressing issue in most intrusion detection systems (IDSs).

Keywords: IEEE 802.11; wireless security; threat detection; machine learning; stacking algorithm

1. Introduction

Modern 5G wireless networks have greatly influenced our lives, allowing for reliable and efficient communication. They are highly valued for their convenience and ever-increasing data transfer capabilities. As more and more of our daily lives depend on safe and reliable access to the Internet through Wi-Fi and 5G cellular networks, security and privacy concerns have begun to grow. In particular, the IEEE 802.11 [1] Wireless Local Area Networks (WLANs) have become the standard way of accessing the Internet in our homes, workplaces, public spaces, and institutions. Since its first adoption, WLAN has been plagued by a multitude of protocol-based and physical layer attacks that aim to interrupt network operation and compromise user privacy.

In recent years, we have witnessed a great growth of interest in machine learning (ML), with impressive feats such as Large Language Models (LLMs). It comes as no surprise that there have been numerous works dedicated to applying advancements in ML techniques to the development of an intrusion detection system (IDS).



Academic Editor: Christos J. Bouras

Received: 23 June 2026

Revised: 26 July 2026

Accepted: 27 July 2026

Published: 29 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Modern IDS suffers from several issues, most notably a high false alarm rate. This may lead to an overwhelming number of alarms being generated by the IDS and dismissed by the network administrator, as the majority of them will be false positives. To address this, this paper proposes a wireless intrusion detection system (WIDS) based on model stacking. Model stacking is an ML technique in which the predictions of several models are analyzed by another model called a meta-classifier to make a more accurate prediction.

Most works involving model stacking utilize shallow learning methods; deep learning (DL) techniques, in particular recurrent neural networks (RNNs) are largely unexplored. For network intrusion detection, the analysis of time patterns is crucial. Therefore, RNN and other DL techniques that allow for recognizing temporal patterns are implemented. In addition, various stacking algorithms are tested and compared for both binary and multiclass classification tasks to find the most universal solution.

Therefore, the main contributions of this paper are as follows:

- Development of a WIDS that leverages multiple ML models to improve system performance.
- Design a heterogeneous stacking framework that integrates sequence-based and sample-level deep learning models to exploit complementary temporal and frame-level information.
- Detailed description of data processing, features selected for the development of the WIDS, and their relevance for the network intrusion detection task.
- Study of the impact of base classifiers' quality on the stacked model performance.
- Comparison of multiple meta-classification strategies, including averaging-based methods, logistic regression, and MLP, for both binary and multiclass intrusion detection.
- An extensive experimental evaluation on the AWID3 dataset and analysis of the impact of different stacking configurations on detection performance and computational complexity.

Unlike previous AWID3 stacking approaches, which primarily combine sample-level classifiers, the proposed framework employs sequence-based deep learning models (LSTM, GRU, and CNN) as base learners, enabling the stacking architecture to exploit temporal dependencies in IEEE 802.11 traffic. The contribution of this work therefore lies in the integration and systematic evaluation of sequence-aware deep learning models within a stacking framework, rather than solely in the achieved classification accuracy.

The paper is structured as follows: Section 2 provides background information on the security of IEEE 802.11 networks and the ML algorithms used for intrusion detection systems. Section 3 surveys the current state-of-the-art solutions for applications of ML algorithms in IDSs. Section 4 details the proposed WIDS architecture. Section 5 describes the development of the WIDS as well as the evaluation of the proposed solution. Finally, Section 6 concludes the work and outlines further directions of research.

2. Background

This section provides background information on WLAN security and ML algorithms used for intrusion detection in wireless networks. Section 2.1 outlines the security mechanisms used in IEEE 802.11, Section 2.2 describes the most common and novel attacks on wireless networks, while Section 2.3 provides information on the ML techniques used in the work.

2.1. WLAN Security

Due to the broadcast nature of wireless communication, Wi-Fi networks are subject to attacks on both the physical and MAC layers, as attackers can capture traffic and inject malicious frames into the network. In response to this, several amendments to the IEEE

802.11 standard were issued. One important characteristic of Wi-Fi networks is that in most modern networks, traffic is encrypted so that the payload of data frames such as IP and TCP headers cannot be accessed and used for intrusion detection. The first security protocol used in WLAN was Wired Equivalent Privacy (WEP), which was discovered to be seriously flawed. The WEP algorithm employs the RC4 stream cipher for encryption, and the Initialization Vectors (IVs) used for computing the encryption key are sent in plaintext. An attacker may eavesdrop for the IVs, and once a number of them have been captured, exploit weaknesses in the RC4 encryption to recover the key.

After the flaws of the WEP protocol became known, new authentication and encryption schemes, Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access 2 (WPA2), were developed as part of the IEEE 802.11i [2] amendment. The main difference between WPA and WPA2 is that WPA uses the Temporal Key Integrity Protocol (TKIP) with RC4 encryption, while WPA2 uses the CCM mode Protocol (CCMP) with the more secure Advanced Encryption Standard (AES) encryption. Due to the vulnerabilities of RC4 encryption, WPA was not intended as a long-term solution. It was meant to replace the faulty WEP protocol in network devices where hardware limitations made it unfeasible to implement AES encryption, as it could be installed with a firmware update. Because AES encryption is more secure, it was designed to address the issues of the insecure WEP protocol. The WPA2 protocol is generally considered robust; however, certain weaknesses have been discovered in the four-way handshake that is used for key exchange in WPA/WPA2. For once, in WPA/WPA2 networks with Pre-Shared Key authentication (WPA/WPA2-PSK), if the PSK is known to the attacker, it is possible to extract the Transmission Key (TK) by capturing the four-way handshake. Another vulnerability of the four-way handshake is that the PSK may be recovered by offline password cracking attacks, such as dictionary attacks.

To address these issues, a new security protocol, Wi-Fi Protected Access 3 (WPA3), was defined in the IEEE 802.11-2020 [3]. The fundamental change in WPA3 is Simultaneous Authentication of Equals (SAE) used for key exchange. Thanks to the changes, it is no longer possible to recover the TK from a captured handshake, and offline brute force and dictionary attacks are impossible, as every attempt to crack the PSK must be done online because of the more robust key exchange algorithm used.

Another enhancement to security was the introduction of Protected Management Frames (PMF) in the 802.11w [4] amendment. Prior to it, management frames, such as Deauthentication, were not encrypted. This made it possible for an attacker to inject fabricated management frames into the network, as they would be received and processed by devices in the network in the same way as benign ones.

2.2. Attack on IEEE 802.11 Wireless Networks

A comprehensive list of attacks on WLAN is provided below:

- **Eavesdropping** is a form of passive attack in which the intruder captures the traffic in the wireless network with the intention of gaining insight into the operation of the network that might be important for the execution of future attacks, such as the number of devices connected to the network and their MAC addresses. The aim of the eavesdropper might also be to retrieve the network key by capturing WEP-encrypted traffic or a WPA/WPA2 handshake.
- **Jamming** is a Denial of Service (DoS) attack on the physical layer. The attacker emits a strong signal in the frequency band the network operates in, reducing Signal-to-Noise Ratio (SNR), either reducing the capacity of the channel, forcing devices in the network to use lower rates, or, when the jamming signal is strong enough, rendering the channel unusable. This attack may lead the station to disconnect from the network and connect to another Access Point (AP) with a stronger signal. That

can be used by the attacker to perform a Man-in-the-Middle (MitM) attack on the now-disconnected devices.

- **Deauthentication Flooding:** The premise of this attack is to disconnect a user or multiple users from the wireless network by sending Deauthentication Management frames with the spoofed MAC address of the AP to which the users are connected. Upon receiving such a frame, a station will disconnect from the network and will have to be authenticated again. The station may initiate an authentication procedure to reconnect with the network; however, it will likely be deauthenticated again. The aim of this attack is DoS, as it effectively prevents any exchange of information in the network. It may also lead to a more sophisticated MitM-type attack, like those described below, because a station disconnected from an AP may attempt to connect to a different network, set up by the attacker to intercept traffic. The execution of such an attack is quite simple, as it only requires a wireless network card capable of frame injection, and there is publicly available software such as MDK3 [5] and Aireplay-ng from the Aircrack-ng [6] suite that can be used, and custom implementation can be easily created using libraries such as Scapy and libtins. A particular kind of Deauthentication Flooding attack is Broadcast Deauthentication Flooding, where the Deauthentication frames are sent to the broadcast MAC address; however, most devices would disregard such frames, rendering the attack of little practical use.
- **Disassociation Flooding,** much like Deauthentication Flooding, is a type of DoS attack that disconnects users from the wireless network. The difference between Deauthentication and Disassociation Flooding is in the frame subtype used; other than that, the attacks are almost identical. Another type of attack is Amok, where the attacker floods the wireless network with both Deauthentication and Disassociation frames.
- **Beacon Flooding** is an attack where the attacker broadcasts a large number of fabricated Beacon Frames advertising non-existing networks. A Beacon frame is a Management-type frame sent periodically by the AP, typically in intervals of 100 ms, used for time synchronization and advertising the presence of the network to clients that may want to connect. The aim of the attack is to confuse users attempting to connect to the wireless network, as they will see a large number of fake networks that cannot be connected to.
- **Association Flooding** is a DoS attack aimed to exhaust the resources of the AP. During the attack, the attacker sends a large number of fabricated Association Request frames. Upon receiving an Association Request frame, the AP begins an authentication procedure and creates an entry in its RAM for the source's MAC address. This may cause memory overload and downgrade or disable the operation of the AP.
- **Reassociation Flooding** is similar in nature to Deauthentication and Disassociation Flooding as it is another DoS attack aimed to disconnect users from the network. A Reassociation Request frame is sent by a station when it is going to associate with another AP within the same Extended Service Set (ESS). This causes the station to disassociate with the previously associated AP and can be exploited to disconnect the user from the network.
- **Request-To-Send/Clear-to-Send Flooding** is a DoS attack that exploits the Request-to-Send (RTS)/Clear-to-Send mechanism. RTS and CTS are Control frames used to reserve the wireless channel for a collision-free transmission of a data frame. Upon receiving an RTS or CTS, all devices in the wireless network must abstain from transmitting any frame in the period specified by the duration field of the CTS frame. An attacker may take advantage of that by constantly sending RTS/CTS frames with

the maximum value of the duration field, thus preventing any other station from using the radio channel, effectively disturbing the network.

- **Probe Request Flooding:** Probe Request and Probe Response are Management frames used for active scanning of Wi-Fi networks. A station searching for available networks sends a Probe Request frame, and the AP responds with a Probe Response frame. During this attack the attacker floods the AP with a large number of fabricated Probe Request frames, causing it to reply with Probe Response Frames and draining the AP's resources.
- **Probe Response Flooding** uses the Probe Request/Response mechanism similarly to Probe Request flooding, however it is targeted at the station not the AP. During this attack the attacker eavesdrop the wireless channel for Probe Request frames; upon capturing one the attacker floods the scanning station with forged Probe Response frames, causing confusion and preventing the victim from connecting to the wireless network.
- **Fake Power Save:** The Power Save mechanism allows power-constrained devices to reduce resource usage and prolong battery life by going into sleep mode. To inform the AP about that, a station would send a Null Frame with Power Management flag set to the AP, which will buffer traffic destined to the station for the duration of sleep mode. This can be exploited by the attacker, who will fabricate frames with Power Management flag set and send them with the source address of the victim. This will cause the AP to buffer data of the client, worsening their connection.
- **Unfair Channel Access:** IEEE 802.11 networks use Carrier Sensing Multiple Access Collision Avoidance (CSMA/CA) as their primary method to provide access to the network for multiple users. To do this, a backoff mechanism is employed. Before sending a frame, a station picks a random number inside a range called the Contention Window (CW) and waits for that number of time slots when the channel is idle; only then may it transmit a frame. In the case of a collision, when two or more stations attempt to transmit in the same time slot, the maximum value of CW is increased to reduce the probability of further collision and the station waits a specified number of time slots again before attempting a retransmission. A malicious user may manipulate their network card to use very low values of CW, abusing the backoff mechanism, hence skipping the waiting for their turn to transmit and disproportionately occupying the channel, preventing other users from utilizing the network.
- **Rogue AP** is a type of attack where the attacker sets up an unauthorized AP that mimics the actual AP by using the same MAC address and Service Set Identifier (SSID) as the genuine AP. If the Rogue AP's signal is stronger than that of the genuine AP, a station might be lured into connecting with the Rogue AP.
- **Evil Twin:** In this attack the attacker mounts a fake AP with open authentication, which may cause devices to automatically connect to an open network. This can be exploited by the attacker to perform a MitM or Captive Portal attack, where the user is redirected to a webpage fabricated by the attacker that asks them for credentials to access the network.
- **Krack**, or Key Reinstallation Attack, is a vulnerability discovered by Vanhoef in reference [7]. The attack exploits a weakness of the four-way handshake used in WPA2-PSK networks, in that the AP will retransmit message 1 or 3 of the handshake if it does not receive message 2 or 4 from the station. This can be exploited by an attacker who deploys a fake AP on a different channel and acts as MitM. The attacker can prevent message 4 of the handshake from reaching the AP, making it resend message 3 of the handshake. Upon receiving the retransmission of message 3, the station will

reinstall an already used key. Certain Android devices were shown to be particularly vulnerable to the attack as they reinstalled an all-zero key.

- **Kr00k** is a vulnerability discovered in certain WiFi devices described in reference [8] that caused them to install an all-zero key after being disconnected from the wireless network. The attacker disconnects a vulnerable station from the network by sending Disassociation frame as in the Disassociation Flooding attack. A device affected by the exploit would then install an all-zero key and proceed to send buffered data using the all-zero key for encryption, despite no longer being associated with the AP. The attacker may capture these frames and decipher the data.

2.3. Machine Learning Algorithms

In recent years, we have witnessed a great surge in the popularity of ML solutions and deep learning (DL) algorithms in the area of cybersecurity and intrusion detection [9]. ML algorithms used in the proposed solution are described below:

- **Neural Network (NN)**, also called an Artificial Neural Network (ANN), is an ML model that consists of multiple layers of units called neurons. One of the most common and basic NN architectures is Multilayer Perceptron (MLP), which is made of multiple fully connected hidden layers. In an MLP the output of a previous layer is input to the next one. The values of each neuron are the sum of the inputs of the layers multiplied by weights calculated during the training process with a constant value, called bias, added to it. The neurons' values are then subjected to an activation function and fed to the next layer. The operation of a layer of neurons can be represented in terms of linear algebra as follows:

$$y_n = A(W_n x_n + b_n) \quad (1)$$

where y_n is the output of the fully connected layer, x_n is the input vector, W_n is the weight matrix, b_n is the bias added to the result of matrix multiplication and A is the activation function.

The NN's weights and biases are learned in the training process. The most common training method is supervised learning, where, opposite to unsupervised learning, data is labeled; that is, for each training sample there is a known true value. In the case of network intrusion detection, traffic instances may be divided into normal and malicious ones. The training data is divided into batches and fed into the network. The model outputs and labels are used to compute the loss function, which measures how much the model's predictions differ from the true values. The purpose of training is to minimize the loss function. It is done by a method called backpropagation that computes the gradient of the loss function in regard to the NN weights. The model's weights are then adjusted according to the gradient, and the process is repeated with a new batch of data. NN are used for their ability to learn and recognize patterns in data in a variety of tasks, such as image and speech recognition, natural language processing and anomaly detection. A multitude of NN architectures have been developed, each specializing in different tasks, making NN applicable to a multitude of tasks across all disciplines.

- **Convolutional Neural Network (CNN)** is a type of NN where, instead of fully connected layers, convolutional layers are used. The principle of operation of a convolutional layer is in the usage of filters instead of individual neurons. The filters are of a fixed size, and have weights assigned to each input. The filter strides along the input data, resulting in a feature mapping, where the outputs are results of applying the filter weights to a portion of the input data.

CNNs are very versatile, in that they can work with one-, two- or even more-dimensional data. The applications of CNN include audio and image processing,

where one-dimensional and two-dimensional CNNs are used, respectively. They are widely used for their ability to extract features from structured data. Another advantage of CNNs is high computational efficiency and a small number of parameters, compared to a fully connected NN.

For network intrusion detection, one-dimensional CNNs are commonly used; the convolution is performed in the time domain. Samples are organized into sequences, and the filters are applied at each timestep. The working of a one-dimensional convolutional layer is presented in Figure 1.

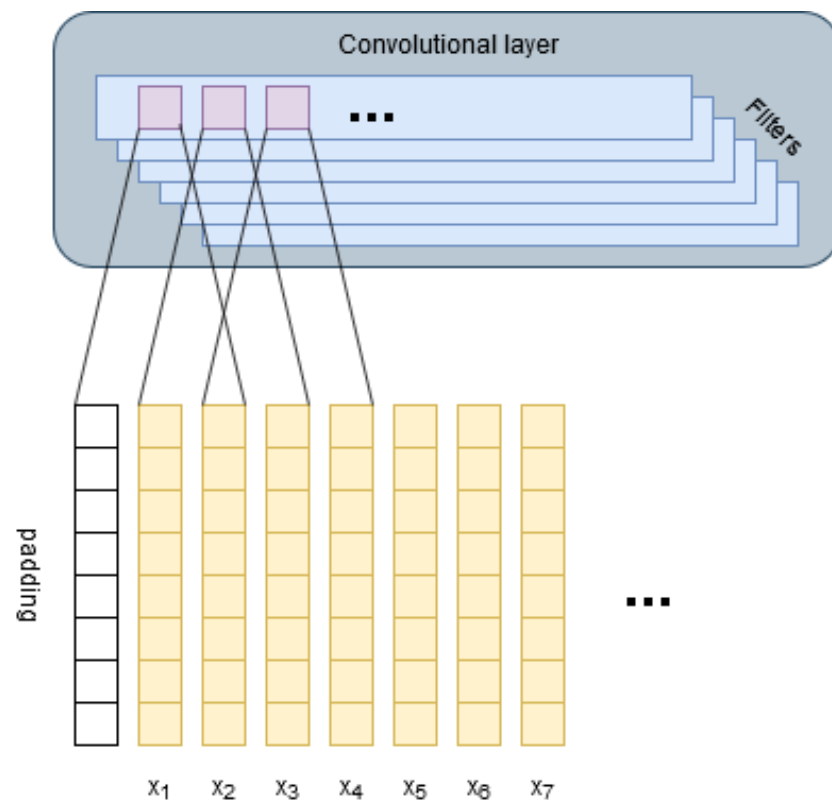


Figure 1. Diagram of a one-dimensional CNN.

- **Long Short-Term Memory (LSTM)** is a type of RNN. These are a kind of NN that is designed to analyze sequential data. In an RNN, the output of the RNN layer is fed as an input in the next timestep, allowing for recognition of temporal patterns. An RNN's weights are trained with a special type of backpropagation algorithm called backpropagation through time. The simple implementations of RNNs suffered from two problems: vanishing and exploding gradients, where, with increasing length of sequences, the gradient of the loss function either rapidly decreased or grew, making the training of an RNN difficult. To address this issue, more elaborate RNN architectures have been developed, namely LSTM and GRU.

RNN are used for tasks where the input data can be represented as a sequence of consecutive samples, such as speech recognition and network intrusion detection. LSTMs excel at modeling long-distance time relationships, especially compared to time-domain CNNs where the context is limited by the filter size.

An LSTM cell can be divided into three parts called gates, namely the forget gate, input gate and output gate; a diagram of an LSTM cell is shown in Figure 2. The LSTM has three inputs: the output of the previous layer x_t in the given timestep, hidden state h_{t-1} and cell state c_{t-1} from the previous timestep. The cell state c_t is used for modeling long-term relationships and can be thought of as a long-term memory of

sorts because it changes little; it is only used internally by the LSTM cell and not passed as an output to the next layer. The hidden state h_t is the short-term memory of the LSTM cell and the actual output of the cell that is fed into the higher layers. The first two gates are the forget gate and input gate that are responsible for updating the previous cell state c_{t-1} ; their operation can be described as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (4)$$

$$c_t = c_{t-1} \odot f_t + i_t \odot \tilde{c}_t \quad (5)$$

where σ denotes the sigmoid function and $\tanh$ hyperbolic tangent is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (7)$$

$\odot$ represents the element-wise vector product, W_n and U_n are weight matrices associated with the input x_t and previous hidden state h_{t-1} and b_n is the bias of a given step. The output of the forget gate f_t is element-wise multiplied with the cell state c_{t-1} . Since the values of the sigmoid function are in the range (0, 1), the result of this operation will be smaller in terms of absolute value, hence the forget gate. The input gate is tasked with extracting the information to be stored in the cell state. The candidate cell state $\tilde{c}_t$ is element-wise multiplied with the output of the input gate i_t and added to the cell state, resulting in the final cell state of current iteration c_t .

The final gate is the output gate, where the final output of the LSTM cell is calculated as follows:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (9)$$

The hidden state h_t is fed forward to the next layer of the model and is passed with cell state c_t to the LSTM cell in the next timestep.

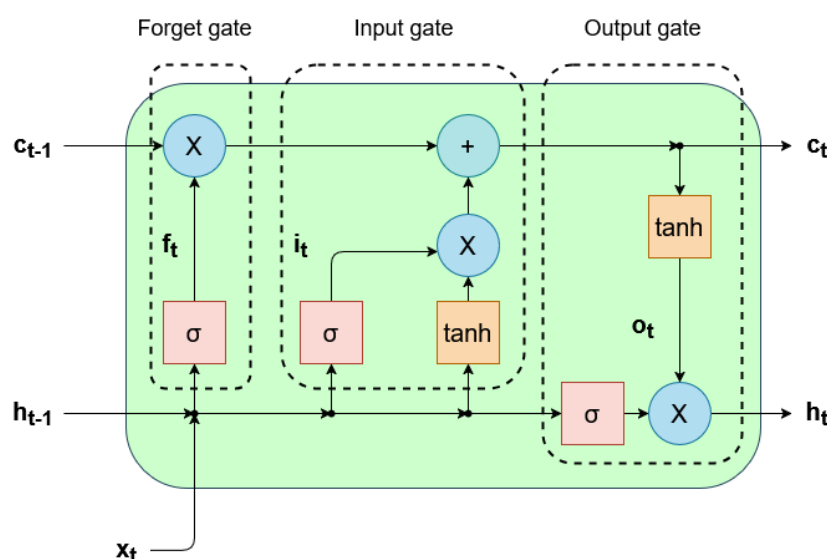


Figure 2. Diagram of an LSTM cell.

- **Gated Recurrent Unit (GRU)** is another RNN architecture and an alternative to LSTM, compared to which it is more lightweight. The fundamental difference between LSTM and GRU is in how the state of the cell is implemented. In LSTM the cell state is separated from the hidden state, whereas in GRU they are one. GRU is made of two gates, the reset gate and the update gate, as shown in Figure 3. The operation of a GRU can be described as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (10)$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (11)$$

$$\hat{h}_t = \tanh(W_h x_t + U_h (h_{t-1} \odot r_t) + b_h) \quad (12)$$

$$h_t = h_{t-1} \odot z_t + \hat{h}_t \odot (1 - z_t) \quad (13)$$

where h_{t-1} is the hidden state of the previous iteration, r_t and z_t are the output of the reset and update gates, x_t is the output of the previous layer, W_n and U_n are weight matrices by which x_t and h_{t-1} are multiplied in respective gates, b_n is the gate bias and h_t is the resulting hidden state, which is also the output of the GRU cell.

The operation of the reset gate is similar to that of the forget gate in LSTM, as its role is to determine how much of the previous hidden state is to be “reset”. The now reset hidden state $h_{t-1} \odot r_t$ is used to calculate the candidate activation vector $\hat{h}_t$, that is, a potential new hidden state of the GRU cell. The update gate is responsible for deciding how much of the previous hidden state h_{t-1} is to be retained and how much is replaced with the candidate activation vector $\hat{h}_t$; the resulting hidden state h_t is a combination of those adjusted by the output of the update gate z_t .

The main advantage of GRU compared to LSTM is better computational efficiency because it involves fewer operations and has a smaller number of parameters, which also makes GRU training faster. However, because of how the cell state is implemented, it is less capable of recognizing patterns in long sequences than LSTM.

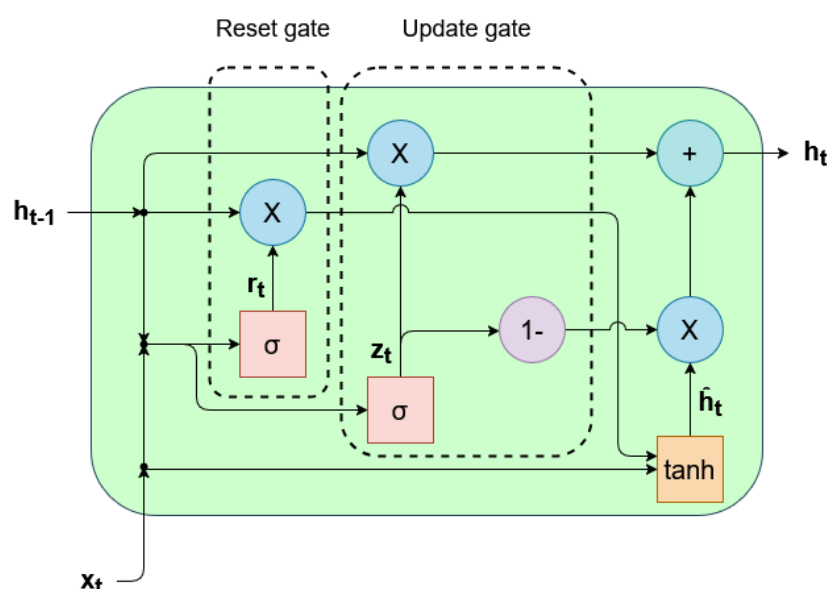


Figure 3. Diagram of a GRU cell.

3. State of the Art

This section reviews existing applications of ML techniques for IDS. It focuses primarily on systems developed for WLANs; however, it also includes other works where the

methodology used was similar or is in other ways relevant to the solution proposed in this paper, even when the authors operate on other datasets for different network types.

In reference [10], a WiFi intrusion detection system is proposed. Several ensemble learning techniques are considered: Bagging, Random Forest (RF), Extra Trees and Extreme Gradient Boosting (XGB). The models were trained on the AWID dataset, and the authors provide a detailed description of processing the data. Of the proposed algorithms, Random Forest was found to be the best-performing one, with an accuracy of 95.87%. Another implementation of ensemble learning is described in reference [11], where the authors used five base classifiers: Support Vector Machines (SVM), RF, XGB, k-Nearest Neighbors (kNN) and MLP. The stacked ensemble model was trained on the AWID3 dataset and showed great improvement. With base classifiers performing at varying levels of accuracy (between 92% and 95% accuracy for the worst and best base classifiers), the stacked ensemble model achieved 98% accuracy, demonstrating the gains of the proposed architecture. The stacked ensemble model is also used in reference [12], again with the AWID3 dataset. The authors used four base classifiers (MLP, RF, XGB and LightGBM) with logistic regression as a meta-classifier. The stacked ensemble model achieved 99.75% accuracy, improving on very robust base classifiers (the best of which had 99.6% accuracy itself), showing that even with very well-performing base classifiers, there is still potential gain with ensemble learning. The authors of [13] developed a rule-based wireless intrusion detection system that integrates a machine learning model for better accuracy. The authors tested several machine learning techniques: Decision Tree (DT), RF, Naive-Bayes Classifier (NBC) and MLP. The models were trained and tested on the AWID3 dataset, and DT was deemed the best, achieving 98.57% accuracy, followed closely by RF with depth 4 at 97.46%.

A real-time WIDS utilizing an ML algorithm is presented in reference [14]. In the proposed system, upon detecting a sudden increase in the volume of traffic, APs will capture the traffic and send it to a WLAN controller, where the traffic will be analyzed using an ML classifier. The authors created their own dataset of attacks on WPA3 Enterprise networks that was used to train ML models. DT, RF, and logistic regression (LR) were compared, and it was found that DT and RF had the highest accuracy of 99.98% and 99.97%, respectively, significantly better than LR. The authors, however, did not provide a description of the data preprocessing employed in the development of the dataset.

RF was implemented in reference [15] for intrusion detection in wireless networks. The author experimented with different feature selection methods and achieved a very high accuracy of 99.999% as well as similarly high other metrics for binary and multiclass classification on the AWID3 dataset. Although the reported performance is very high, the methodology makes it difficult to assess the contribution of the selected features. The paper employs more than 140 input features but does not provide the complete feature list or sufficient details of the feature selection procedure. Consequently, the reproducibility of the reported results and their generalizability to independent traffic captures cannot be fully evaluated. Furthermore, the use of a very large feature set increases the possibility of including capture-specific attributes that may inadvertently introduce dataset bias.

NN is developed in reference [16] for intrusion detection in wireless networks. The model was trained on the AWID3 dataset and achieved 99.5% accuracy; however, other metrics were less impressive, with 95.2% precision, 94.5% F1 score and 94.0% recall.

Among the most promising and widely adopted ML approaches are CNNs, which have proven successful in network traffic analysis and anomaly detection. Ref. [17] describes a CNN model with impressive 99.55% accuracy and a very low false positive rate of 0.4% for binary classification and 0.12% for multiclass classification, respectively.

In reference [18] the authors propose an IDS based on a hybrid architecture comprising CNN and Bidirectional LSTM (BiLSTM), designed for IoT devices with limited resources. The usage of CNN provided the model with efficiency and low computational resource and memory utilization, making it suitable for IoT applications, while BiLSTM achieved an accuracy of 97.28% for binary and 96.91% for multiclass classification. An IDS utilizing 1D-CNN is developed in reference [19]. The model was trained on the AWID dataset and achieved 94.6% and 95.5% accuracy, 94.6% and 97.2% precision, 95.1% and 93.4% recall for binary and multiclass classification, respectively.

Another implementation of CNN is presented in reference [20], where the Gramian Angular Field (GAF) is used to transform data from the AWID3 dataset into images that are then passed to the CNN. The best-performing model of the ones tested was 2D-CNN with two convolutional layers that achieved 99.75% accuracy. Notably, it outperformed 1D-CNN which had a vector of the features that were used to create the GAF images as input, proving the efficiency of the proposed method.

The authors of [21] developed an IDS implementing Fully Convolutional Network (FCN), LSTM, and Vision Transformer (ViT). The FCN was used for its ability to extract local features, which are then passed to the Transformer Encoder of ViT as it better analyzes long-distance spatial features. The output of ViT is fed to LSTM for its ability to recognize temporal patterns. The model was trained on the AWID dataset and achieved an impressive 99.973% accuracy.

In reference [22], CNN is used to extract features from the AWID3 dataset and is compared with statistical measures. The features extracted by both methods are then used for training ML models: kNN, SVM, and Gradient Boosting Trees (GBT). Of the six experiments conducted, GBT with features extracted by statistical measures was the best-performing, with 97.9% accuracy. The feature selection method based on statistical measures provided better results for all the tested classifiers than feature extraction by CNN.

A novel strategy called Hybrid Regression-Classification (HRC) is presented in reference [23]. It aims to reduce the effects of an imbalanced dataset (AWID3 was used) by employing two ML algorithms—a regressor and a classifier. A CNN1D-LSTM regressor is trained on normal traffic samples to predict future samples. If the difference between predicted and actual traffic is greater than an established threshold, the sample is deemed anomalous, and the type of attack is classified by the CNN1D multiclass classifier. The hybrid model achieved 91% accuracy and recall over 85% for all categories. The results were compared with the Bayesian approach and histogram gradient boosting. These methods performed significantly worse in terms of detecting attacks, reaching a 0% F1 score in attack categories, proving the effectiveness of HRC when dealing with an imbalanced dataset. Authors also tested HRC with the NSL-KDD dataset, and they reached a higher accuracy of 97%.

The authors of [24] implemented a weighted stacking classification algorithm for intrusion detection. In this study, RF, XGB, and kNN were chosen as base classifiers, and LR was used as a meta classifier. The proposed approach was implemented on NSL-KDD and CIC-IDS2018 datasets, resulting in 87.44% and 99.87% accuracy, respectively, and the model stacking ensemble method improved the performance of the classifier.

Ref. [25] implements Stacked Autoencoder (SAE) to extract the most relevant features. Afterwards, the five most important features were selected from the AWID dataset, and the features extracted by the SAE were used to train the SVM, resulting in a model that requires few computational resources, suitable for implementation on IoT devices. This approach achieved an accuracy of 98.22%, a detection rate of 97.64%, and a false alert rate of 1.20%.

In reference [26], Secure Deep Neural framework is proposed for use in an SDN controller. The traffic is divided into time frames that are then analyzed by LSTM to determine if an attack has occurred. When an attack is detected, the AP will drop all traffic coming from the attacker's MAC address. The model was trained on the AWID dataset, and the proposed framework was evaluated in the mininet-wifi emulator, achieving 97.8% accuracy.

In reference [27], a federated learning approach is presented. In the proposed system, devices operating in the network train local ML models using decentralized datasets. Then, the trained models are sent to a central server, where the weights of the local models are aggregated by averaging them to assemble the weights of a global ML model. Next, the global model is sent back to the devices for the next cycle of training. Such an approach allows for improved data privacy because only ML model weights are shared, not the data on which the models train. The AWID dataset was also used to train Stacked Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN) for binary classification. The model reached 99% accuracy outperforming other federated learning models and classic ML algorithms such as RF and SVM, the other metrics were 99.3% for precision, 99% recall, 98.9% F1 score and 99.5% detection rate. The authors also showed that the performance of the model improved as more devices participated in the federated learning process.

In reference [28], the authors study the transferability of the ML model for intrusion detection. Two architectures, MLP and CNN, are first trained on the AWID2 dataset, and then their performance is assessed on the AWID3 dataset. This has shown a significant decrease in model accuracy (from 99.84% to 96.52% for MLP and from 99.94% to 97.28% for CNN), indicating that model performance on a dataset may not be adequate for real-world implementation.

A summary of papers exploring applications of ML techniques for threat detection in Wi-Fi networks, the ML architectures used, and their performance metrics is presented in Table 1. Model stacking has been shown to greatly improve the accuracy of ensemble models; however, in intrusion detection, mostly Shallow Learning techniques, such as tree classifiers, have been utilized so far. This work aims to implement more advanced ML architectures, such as RNN and CNN, which have been shown to achieve superior metrics in the stacked WIDS, to investigate the possibilities of ensemble learning. Unlike previous studies, which typically report the performance of a single stacking configuration, our work systematically investigates the impact of different meta-classification strategies and heterogeneous base classifier combinations, providing practical design guidelines for stacking-based Wi-Fi intrusion detection systems. The primary contribution of this work is therefore not the introduction of a new deep learning model, but the design and evaluation of a practical wireless intrusion detection system that combines appropriate feature engineering, preprocessing, sequence construction, and model optimization for IEEE 802.11 traffic. We therefore argue that the usage of state-of-the-art well established model architectures is appropriate, as it shows that model stacking can be used in combination with them to improve the detection accuracy.

Recent research has also focused on improving wireless security and reliability at the physical layer. In particular, advanced channel estimation techniques for near-field XL-MIMO systems, such as enhanced polar-domain channel estimation under low-SNR conditions, enable more accurate reconstruction of propagation characteristics and improve communication robustness. Although these approaches operate at a different layer of the protocol stack than the proposed WIDS, they are complementary rather than competing technologies. The proposed intrusion detection framework relies exclusively on IEEE 802.11 MAC-layer traffic features with selected PHY layer elements placed in the Radiotap Header available in the AWID3 dataset and therefore does not exploit full physical-layer

information such as channel state information (CSI). Nevertheless, combining physical-layer measurements with sequence-aware deep learning models represents a promising direction for future wireless intrusion detection systems.

Table 1. Comparison of related works on Wi-Fi threat detection using machine learning techniques.

Refs.	Year	Methods	Datasets	Performance
[10]	2018	ensemble learning	AWID	ACC = 95.87%
[26]	2019	LSTM	AWID	ACC = 97.8%
[13]	2023	DT	AWID3	ACC = 98.57%
[14]	2022	DT	Custom dataset	ACC = 99.98%
[17]	2022	CNN	CIC-IDS2017	ACC = 99.55% DR = 99.63% FAR = 0.12%
[25]	2022	SAE and SVM	AWID	ACC = 98.22% DR = 97.64% FAR = 1.20%
[24]	2022	weighted stacking, RF, XGB and kNN	NSL-KDD, CIC-IDS2018	ACC = 87.44% for NSL-KDD; ACC = 99.87% for CIC-IDS2018
[16]	2023	NN	AWID3	ACC = 99.5%. F1 = 94.5%
[19]	2023	1D-CNN	AWID	ACC = 94.6% DR = 94.6% REC = 95.1%
[20]	2023	GAF and CNN	AWID3	ACC = 99.75% DR = 99.61% REC = 99.63%
[21]	2023	FCN, ViT and LSTM	AWID	ACC = 99.973%
[15]	2024	RF	AWID3	ACC = 99.999%
[18]	2024	CNN-BiLSTM	UNSW-NB15	ACC = 96.91% F1 = 97.33%
[22]	2024	GBT	AWID3	ACC = 97.9%
[23]	2024	HRC, CNN1D-LSTM and CNN1D	AWID3, NSL-KDD	ACC = 91% for AWID3, ACC = 97% for NSL-KDD
[27]	2024	DBN-RBM	AWID	ACC = 99%, DR = 99.5%, F1 = 98.9%
[28]	2025	CNN and MLP	AWID, AWID3	ACC = 97.28% for CNN and ACC = 96.52% for MLP
[11]	2025	stacked ensemble learning	AWID3	ACC = 98%
[12]	2025	stacked ensemble learning	AWID3	ACC = 99.75%
This work	2026	stacked ensemble learning based on DL	AWID3	ACC = 99.88% and F1 = 99.51%

4. Proposed Solution

This section provides insight into the operation of the proposed WIDS. Section 4.1 details the system architecture, and Section 4.2 describes the algorithms used to construct stacked models.

4.1. System Architecture

Model stacking is an ensemble learning technique in which a prediction is made based on the outputs of several machine learning models called base classifiers. To avoid data leakage, out-of-fold predictions are produced using K-fold cross-validation, where each base model is trained on a subset of the training data and evaluated on held-out folds.

These out-of-fold predictions are then used as input features to train a meta-model called a meta-classifier, which learns how to combine the base models' outputs to improve final prediction accuracy. A diagram of the proposed system architecture can be seen in Figure 4. The main premise of model stacking is leveraging the ability of individual ML models to perform better with various data, thereby improving system robustness. In cases where a single model fails to make an accurate prediction, the meta-classifier might still be able to make a correct decision, improving overall system performance.

Another advantage of the proposed approach is its flexibility. Base classifiers can be implemented using different machine learning models, as demonstrated in this work by combining recurrent neural network architectures with models operating on individual network frames. This enables the framework to capture both temporal dependencies and frame-level characteristics of network traffic. Furthermore, lightweight aggregation methods, such as logistic regression and shallow MLPs, require relatively short training times, facilitating rapid adaptation of the stacking model. Consequently, the framework allows different stacking configurations to be selected depending on the current workload and performance requirements. For example, during periods of lower traffic intensity, a configuration employing a larger number of base classifiers may be preferred to maximize detection performance, whereas under higher traffic loads a simpler stacking configuration may be selected to reduce computational overhead.

In addition to stacking, simple ensemble averaging of base classifier outputs can also be explored as a computationally efficient alternative. Unlike stacking, this approach does not require training a meta-classifier, as final predictions are obtained by aggregating the outputs of independently trained base models. This makes it suitable for real-time or resource-limited IDS deployments.

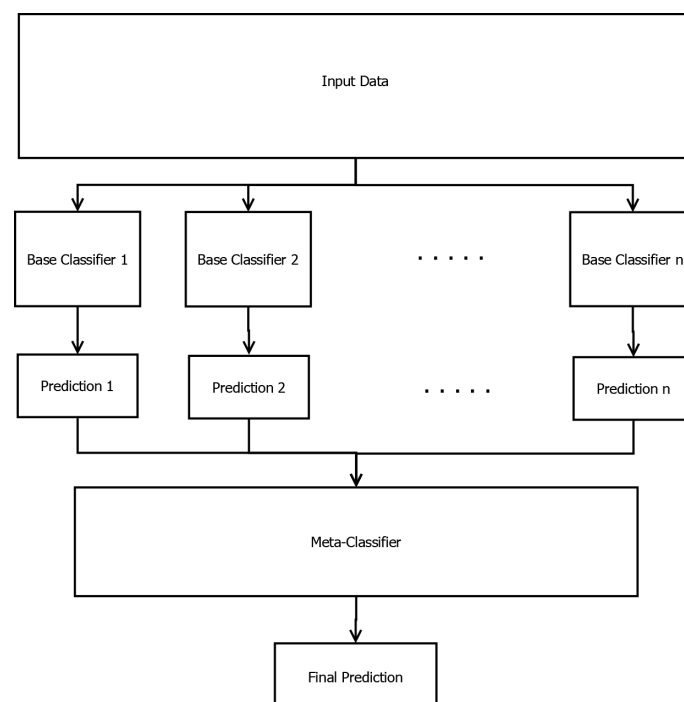


Figure 4. Proposed system architecture.

4.2. Stacking Algorithms

In the experiment, several methods of aggregating the base classifiers' outputs are investigated: arithmetic mean, F1 score weighted mean, regression, and MLP. Each of the stacking algorithms is used for both binary and multiclass classification.

The first and simplest of the proposed methods is the arithmetic mean. The output of each of the base classifiers is added, and the sum is divided by the number of classifiers. In the case of multiclass classification, the outputs are averaged for each of the classes.

The second aggregation strategy is an F1 score weighted average. The weight assigned to each base classifier reflects its predictive performance estimated during the training stage. To avoid information leakage, the weights are computed exclusively from the F1 scores obtained using the out-of-fold predictions generated during the K-fold cross-validation procedure described for the stacking framework. Consequently, the independent test set is not used for determining the weights and is reserved solely for the final performance evaluation. Since the F1 scores of the base classifiers are typically very high (often exceeding 0.9), using them directly would result in only minor differences between the assigned weights. Therefore, a log-odds transformation is applied to increase the separation between high-performing classifiers while preserving their ordering [24]:

$$w_i = \log\left(\frac{F1_i + \varepsilon}{1 - F1_i + \varepsilon}\right) \quad (14)$$

where w_i denotes the weight assigned to the i -th base classifier, $F1_i$ is the F1 score estimated from the out-of-fold validation predictions generated during K-fold cross-validation, and ε is a small positive constant introduced to avoid numerical singularities for the limiting cases $F1 = 0$ and $F1 = 1$. In this work $\varepsilon = 10^{-8}$. The weights are subsequently normalized so that their sum equals one. In the stacked model, the predictions of base classifiers are multiplied by their assigned weights and summed to obtain the final prediction.

Both averaging-based aggregation methods are computationally inexpensive, as they do not require training an additional meta-classifier. Consequently, they can be readily applied to any combination of base classifiers.

The next stacking algorithm implemented is regression. It is a simple machine learning model consisting only of an output layer. For binary classification, the output layer consists of a single neuron with sigmoid activation, called logistic regression, while for multiclass classification, it contains three neurons, one for each class, with softmax activation. The regression stacked classifier is presented in Figure 5.

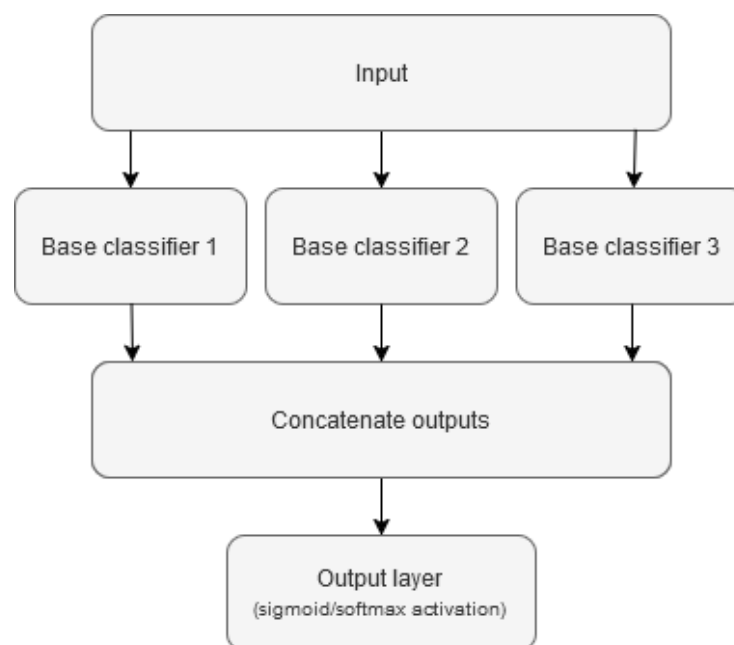


Figure 5. Stacked model with regression meta-classifier.

The regression model serves as the meta-classifier. It receives the outputs of the base classifiers and learns how to combine them into the final prediction. Its training procedure is summarized in Algorithm 1.

Algorithm 1 The Training Procedure of the Proposed Stacking Framework

```

1: Input: Base classifiers  $\mathcal{B}$ , aggregation method  $\mathcal{A} \in \{\text{Arithmetic Mean, F1 Weighted Mean, Logistic Regression, MLP}\}$ , training dataset  $\mathcal{D}$ , number of folds  $K$ 
2: Output: Trained stacked classifier
3: Split  $\mathcal{D}$  into  $K$  folds
4: Initialize out-of-fold prediction set  $\mathcal{P} \leftarrow \emptyset$ 
5: Initialize validation F1 score set  $\mathcal{F} \leftarrow \emptyset$ 
6: for each base classifier  $B_i \in \mathcal{B}$  do
7:   Initialize prediction vector  $P_i$ 
8:   for  $k = 1$  to  $K$  do
9:      $\mathcal{D}_{train} \leftarrow \mathcal{D} \setminus \text{Fold}_k$ 
10:     $\mathcal{D}_{valid} \leftarrow \text{Fold}_k$ 
11:    Train  $B_i$  on  $\mathcal{D}_{train}$ 
12:    Predict labels for  $\mathcal{D}_{valid}$ 
13:    Store predictions in  $P_i$ 
14:   end for
15:   Store  $P_i$  in  $\mathcal{P}$ 
16:   Compute the out-of-fold F1 score of  $B_i$  using the validation predictions
17:   Store the obtained F1 score in  $\mathcal{F}$ 
18:   Retrain  $B_i$  using the complete training dataset  $\mathcal{D}$ 
19: end for
20: Rank the base classifiers according to the out-of-fold F1 scores
21: Select the predefined subset of base classifiers for stacking
22: if  $\mathcal{A}$  is Arithmetic Mean then
23:   Construct the stacked classifier using arithmetic averaging
24: else if  $\mathcal{A}$  is F1 Weighted Mean then
25:   Compute classifier weights from  $\mathcal{F}$  using Equation (14)
26:   Normalize the classifier weights
27:   Construct the stacked classifier using F1 weighted averaging
28: else
29:   Construct the meta-training dataset from the out-of-fold predictions  $\mathcal{P}$ 
30:   Train the meta-classifier  $\mathcal{A}$  using the meta-training dataset
31:   Construct the stacked classifier using the trained meta-classifier
32: end if
33: return the trained stacked classifier

```

To train the meta-classifier, a K-fold cross-validation procedure is used. The training dataset is divided into K folds (in our case $K = 5$). For each fold, every base classifier is trained on the remaining K-1 folds and generates predictions for the held-out fold. In this way, every training sample receives predictions from models that have not been trained on that sample. These out-of-fold predictions are then collected and used as features for training the meta-classifier, while the original class labels are retained as targets. After the out-of-fold predictions have been generated, each base classifier is retrained on the entire training dataset. The final stacked model is constructed by connecting the outputs of the retrained base classifiers to the inputs of the meta-classifier.

The final aggregation strategy employs a lightweight multilayer perceptron (MLP) as the meta-classifier. It consists of two hidden layers with four neurons each and ReLU activation functions. The output layer is identical to that of the regression meta-classifier: one neuron with sigmoid activation for binary classification and three neurons with softmax activation for multiclass classification, as shown in Figure 6. Similar to logistic regression, the MLP is trained using the out-of-fold predictions generated during K-fold cross-validation.

This ensures that the meta-classifier learns from unbiased predictions and reduces the risk of overfitting.

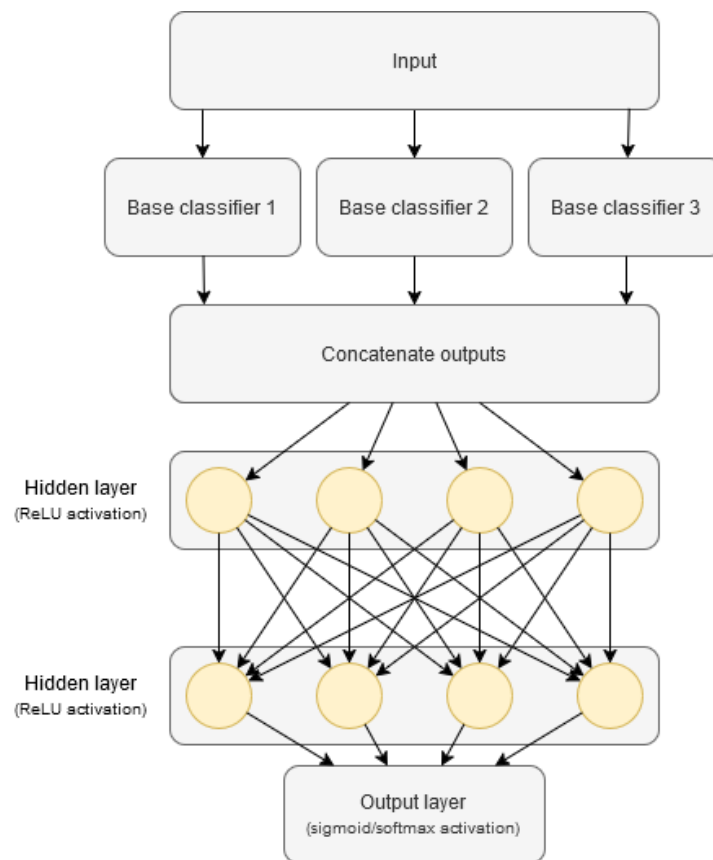


Figure 6. Stacked model with MLP meta-classifier.

Both meta-classifiers use the Adam optimizer with default parameters. Binary focal cross-entropy is used as the loss function for binary classification, while categorical focal cross-entropy is used for multiclass classification. Because both meta-classifiers contain only a small number of trainable parameters, they introduce only a negligible computational overhead during both training and inference. Consequently, the overall computational complexity remains dominated by the base classifiers.

5. System Development and Evaluation

This section focuses on the development and evaluation of the proposed WIDS. Section 5.1 describes the dataset used for the experiment, as well as data-related tasks, such as feature selection and data preprocessing. Section 5.4 defines the metrics used for evaluating model performance. Section 5.5 details the architecture of base classifiers, which are then used to construct the stacked WIDS for binary and multiclass classification in Section 5.6 and Section 5.7, respectively.

5.1. Data

The publicly available datasets including attacks on IEEE 802.11 are Aegean Wi-Fi Intrusion Dataset 2 [29] (AWID2) and AWID3 [30]. The AWID2 dataset is older and somewhat dated, as it includes attacks on the WEP protocol, which is long deprecated. Therefore, for the training of the models and evaluation of the WIDS AWID3 is used, as it contains traffic samples from a modern WPA2 network and includes more novel attacks, such as Krack.

5.2. Dataset Description

The AWID3 dataset consists of 13 captures of wireless traffic, each containing a different attack on the network as well as normal traffic. The dataset is available both in CSV and pcap format. Of the 13 files that make up the dataset, 7 were chosen that focus on attacks on the second layer of IEEE 802.11 networks. Attacks specific to higher layers, such as SQL Injection and Website Spoofing, were left out. The chosen files and attacks they include are shown in Table 2. As is apparent, the dataset is heavily imbalanced because malicious traffic makes up only a small fraction of all traffic. The Rogue AP and (Re)Assoc attacks especially make up a very small percentage of their respective files at only 0.07% and 0.29% of the total samples.

Before any preprocessing, the dataset was divided into independent training and test subsets. All subsequent preprocessing steps were designed using the training data only and then applied unchanged to the test data to prevent information leakage. To address class imbalance, for models that take single samples as input, a balanced dataset was created by downsampling the majority normal class to match the number of attack samples. The balancing procedure was performed only after the training and test subsets had been separated. Consequently, no samples from the test set were used during the balancing process. For RNN models, such an approach was impossible, as these require data to be in order to work properly. To deal with it, the dataset was divided into non-overlapping sequences, and only those that included any attack samples were kept. The percentage of attack samples increased as the length of the sequences decreased, as shown in Table 3. A sequence length of 64 was chosen because it offered a good balance of class distribution, a length of sequences that allowed the RNN models to learn temporal patterns, and a smaller total number of samples, making the model training faster. The described approach was suitable for RNN and CNN model, as it improves the data balance, whilst preserving the temporal patterns present in the data. Importantly, the models perform predictions at the individual sample level rather than classifying entire sequences. Consequently, even the selected sequences remain dominated by normal samples, allowing the models to learn both benign and malicious behavior while preserving the temporal relationships necessary for sequence modelling.

Table 2. Attacks on Wi-Fi networks present in AWID3 dataset and number of frames in files.

File Name	Attacks	Total Number of Frames	Number of Attack Frames	%
Deauth	Deauth. Flooding Deauth. Flooding Broadcast	1,626,472	38,945	2.39
Disass	Disass. Flooding Disass. Flooding Broadcast Amok	2,013,719	75,131	3.73
(Re)Assoc	Assoc. Request Flooding Reassoc. Request Flooding Beacon Flooding	1,843,939	5503	0.29
Rogue_AP	Rogue AP	1,973,193	1310	0.07
Krack	Channel MitM attack Key Reinstallation	1,438,493	49,990	3.47
kr00k	TK Reinstallation	2,900,458	191,803	6.61
Evil_Twin	Captive Portal Man in the middle	3,778,728	102,059	2.70

Table 3. Distribution of attacks in sequential dataset.

Sequence Length	Total Samples	Attack Samples	Attack %	Normal Samples	Normal %
1024	6,114,304	467,317	7.6%	5,646,987	92.4%
512	5,444,096	467,317	8.6%	4,976,779	91.4%
256	4,679,168	467,317	10.0%	4,211,851	90.0%
128	3,965,444	467,317	11.8%	3,497,227	88.2%
64	3,356,032	467,318	13.9%	2,888,714	86.1%
32	2,869,376	467,318	16.3%	2,402,058	83.7%
16	2,463,088	467,318	19.0%	1,995,770	81.0%
8	2,022,816	467,318	23.1%	1,555,498	76.9%
6	1,818,600	467,318	25.7%	1,351,282	74.3%
4	1,438,456	467,318	32.5%	971,138	67.5%

5.3. Feature Selection and Data Preprocessing

The authors of the AWID3 dataset proposed a feature selection method in reference [31] as seen in Table 4. The chosen features are related only to the first and second layers of IEEE 802.11 WLAN. Additionally, they are applicable to every frame, as those used only for a specific frame type or subtype were omitted.

Table 4. Features chosen by Chatzoglou et al. in reference [31] and their preprocessing.

Feature	Preprocessing
frame.len	Min–Max Scaling
radiotap.len	Min–Max Scaling
radiotap.dbm_antsignal	Min–Max Scaling
wlan.duration	Min–Max Scaling
radiotap.present.tsft	One-Hot Encoding
radiotap.channel.freq	One-Hot Encoding
radiotap.channel.type.cck	One-Hot Encoding
radiotap.channel.type.ofdm	One-Hot Encoding
wlan.fc.type	One-Hot Encoding
wlan.fc.subtype	One-Hot Encoding
wlan.fc.ds	One-Hot Encoding
wlan.fc.frag	One-Hot Encoding
wlan.fc.retry	One-Hot Encoding
wlan.fc.pwrmtgt	One-Hot Encoding
wlan.fc.moredata	One-Hot Encoding
wlan.fc.protected	One-Hot Encoding

The features selected by the authors, as well as the preprocessing applied to them, can be seen in Table 5. For the training of the base classifiers, similar feature selection and preprocessing were used, with the following differences:

- frame.time_delta was included, as it is a key factor in temporal patterns.
- binary features such as Radiotap Header and IEEE 802.11 Frame Control flags were left unchanged, not One-Hot Encoding (OHE).
- The authors of [31] made the decision to OHE channel frequency. This approach was feasible because there are few frequencies in the data set. To reduce the possible number of features and provide a more general method that will be compatible with all channels in the 2.4 GHz and 5 GHz spectrum as well as other datasets, the following is proposed:
 - Two features by the names of 2ghz_spectrum and 5ghz_spectrum were created to indicate in which band a frame was sent.

- The frequency was Min–Max-scaled with the lowest and highest channel frequency in the given band used as the range.

The features chosen for the experiment, their preprocessing and values can be seen in Table 5. The categorical features were OHE, and for numerical features, Min–Max scaling was applied. To determine the scaling parameters and identify outliers, the interquartile range (IQR) was used. In order to preserve the data's temporal continuity, frames containing outlier feature values were clipped to the IQR bounds. The meaning of features and their relevance to the intrusion detection task is explained below:

- `frame.len`—the total length of the frame in bytes. It may be indicative of an attack, as, for example, a fake AP may not include optional parameters in the Beacon frames it emits; therefore, they would be shorter than those of the genuine AP, allowing for detection of such an attack.
- `frame.time_delta`—time interval from the previous frame in seconds. It is crucial for modeling temporal relationships between frames. A large number of similar frames, sent in short time intervals, might be a signal of a flooding attack that takes place.
- `radiotap.len`—length of the Radiotap header in bytes. The Radiotap header is included in captures of IEEE 802.11 traffic and contains information about the physical layer, such as channel frequency and power of the received signal. A longer Radiotap header indicates inclusion of additional fields that may be characteristic for frames sent by a specific station.
- `radiotap.dbm_antsignal`—power of the received signal in dBm. A difference in received signal power may help detect attacks in which the attacker impersonates a station or AP in the network, like a Rogue AP, as the signal power will differ from the genuine user. As can be seen in Table 5, the values are significantly smaller than what one may expect in a wireless network. That is because the Radiotap header includes three numerical values—one for each of the two antennas, as the authors of the dataset used a wireless card with two antennas for capturing the network traffic, and one combined value, i.e., the total power on the two antennas. In the CSV files of the dataset, the three values have been added together; hence, the values are about three times lower than what would be expected.
- `wlan.duration`—is a mandatory field of the IEEE 802.11 header. It is used in the CSMA/CA mechanism to inform other stations how long the frame will be transmitted to ensure collision free transmission.
- `wlan.fc.type`—is the type of the frame contained in the Frame Control field of the IEEE 802.11 header. All frames are one of the three types: Management (type 0), Control (type 1) or Data (type 2). The role of Management frames is, among others, time synchronization, managing access by authenticating clients and advertising the network. Control frames are responsible for control of transmission in the wireless channel; they are used for reserving the wireless channel for a contention-free transmission and acknowledgement of receiving other frames. Data frames are frames that contain data payloads sent across the network.
- `wlan.fc.subtype`—in combination with frame type, frame subtype identifies a specific class of frames and their role in the network. A frame subtype alone is not enough to identify its function, as frames of different types can have the same subtype; for example, both Deauthentication and CTS frames are of subtype 12. However, they are of different types and therefore are distinct. The pair of frame type and subtype is fundamental in IDS, because certain attacks are very closely associated with a particular frame type and subtype.
- `wlan.fc.ds`—a Distribution System (DS) is a network segment that the wireless network is connected to, usually the AP connection to the Internet Service Provider. The Frame

Control field of an IEEE 802.11 header contains the FromDS and ToDS flags that indicate in which direction a frame was sent, most commonly from AP to station (FromDS) or vice versa (ToDS). This is significant in detecting MitM-type attacks, because more often than not, only the AP would send frames with the FromDS flag set.

- radiotap.present.tsft—indicates presence of Time Synchronization Function Timer in the Radiotap header
- radiotap.channel.type.cck—flag of the Radiotap header, indicates whether the frame was sent using Complementary Code Keying (CCK) modulation, mainly used in IEEE 802.11b
- radiotap.channel.type.ofdm—flag of the Radiotap header, indicates whether the frame was sent using Orthogonal Frequency Division Multiplexing (OFDM) modulation
- wlan.fc.frag—fragmented flag in Frame Control of IEEE 802.11 header, informs whether the frame is being transmitted in fragments.
- wlan.fc.retry—if the retry flag is set, the frame is a retransmission of a previous frame that was not correctly received or acknowledged by the destination.
- wlan.fc.pwrmtg—the power management flag is used when the station enters power save mode, during which the station is idle and will not send nor receive any frames.
- wlan.fc.protected—the protected flag informs whether a frame is protected using encryption. When PMF is used, genuine management frames will have the protected flag set, whereas the fabricated ones will not.
- 2ghz_spectrum/5ghz_spectrum/freq—describe the frequency of the wireless channel the frame was sent in. It can be used to detect side channel MitM attacks.

Table 5. Features chosen, their type, preprocessing and values.

Feature	Type	Preprocessing	Values
frame.len	numeric	Min–Max Scaling	from 70 to 3220
frame.time_delta	numeric	Min–Max Scaling	from 0 to 0.001817
radiotap.len	numeric	Min–Max Scaling	from 48 to 64
radiotap.dbm_antsignal	numeric	Min–Max Scaling	from −255 to −78
wlan.duration	numeric	Min–Max Scaling	from 0 to 726
wlan.fc.type	categorical	One-Hot Encoding	0, 1, 2
wlan.fc.subtype	categorical	One-Hot Encoding	from 0 to 15
wlan.fc.ds	categorical	One-Hot Encoding	0, 1, 2, 3
radiotap.present.tsft	binary	convert to 0, 1	‘0-0-0’, ‘1-0-0’
radiotap.channel.type.cck	binary	None	0, 1
radiotap.channel.type.ofdm	binary	None	0, 1
wlan.fc.frag	binary	None	0, 1
wlan.fc.retry	binary	None	0, 1
wlan.fc.pwrmtg	binary	None	0, 1
wlan.fc.moredata	binary	None	0, 1
wlan.fc.protected	binary	None	0, 1
2ghz_spectrum	binary	None	0, 1
5ghz_spectrum	binary	None	0, 1
freq	numeric	Min–Max scaling	from 0 to 1

The dataset in CSV format was used, preprocessed and converted into tfrecords format. The few present records with missing values were removed. Each of the wireless traffic captures in the AWID3 dataset in CSV format is divided into CSV files of 50,000 records (frames), representing continuous data. The training, validation and test datasets were split on a per-file basis to limit information leakage between them while ensuring that each of the attack scenarios is present in all of the datasets. All preprocessing parameters,

including feature scaling statistics, were estimated exclusively from the training data and subsequently applied to the test set without recalculation. This ensured that no information from the test set influenced model training.

For the purpose of multiclass classification, the data was divided into three classes as shown in Table 6. Attacks that are similar in their nature were assigned to the same class; for example, Deauthentication and Disassociation flooding include sending a large number of their respective frame subtypes. Therefore, they were included in the same class.

Table 6. Attacks in the AWID3 dataset divided into classes.

Class	Label	Attacks
Normal	0	N/A
Flooding	1	Deauthentication, Disassociation and (Re)Association Flooding, Kr00k
Impersonation	2	Rogue AP, Krack, Evil Twin

5.4. Metrics

For the evaluation of the system performance, the metrics of accuracy, precision, recall, and F1 score are used [32]. They are the most common metrics used to assess classifier performance, as they provide a holistic view of how ML models perform.

The first metric is accuracy, which represents the percentage of correct predictions. It can be expressed as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (15)$$

where TP is the total number of true positives, that is, positive samples that the ML model has predicted correctly. Analogously, TN stands for true negatives, negative samples predicted correctly. FP is the number of false positives, that is, negative samples that were incorrectly predicted as positive. FN represents false negatives, the number of positive samples that were predicted to be negative. Accuracy is the primary metric used to evaluate the performance of a classifier; however, it does not provide all information about how the model performs, as it does not identify the mistakes the model makes. For example, when using a heavily imbalanced dataset, a model can achieve high accuracy by being biased towards the majority class. Therefore, additional evaluation metrics are necessary.

Precision is defined as

$$precision = \frac{TP}{TP + FP} \quad (16)$$

informs how trustworthy positive predictions of the classifier are. It is crucially important in the context of an IDS, as malicious traffic makes up only a small percentage of all the traffic in the network. Therefore, using an IDS with low precision will result in a large number of false alarms, which is a common issue in the development of such systems.

Recall is defined as

$$recall = \frac{TP}{TP + FN} \quad (17)$$

measures how sensitive the classifier is to positive samples. In terms of an IDS, recall signifies the fraction of attacks that were detected. Using an ML with low recall will result in undetected attacks, hence compromising the security of the network.

F1 score is a metric that combines precision and recall. It can be expressed in terms of true positives, false positives, and true negatives, or precision and recall:

$$F_1 = \frac{2TP}{2TP + FP + FN} = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (18)$$

The F1 score provides a good indication of overall classifier performance, as it leverages both precision and recall. Therefore, it is used as the main metric to compare models in the later sections. In the case of multiclass classification, recall, precision, and F1 score were calculated for each of the three classes and averaged.

5.5. Base Classifiers Performance

A total of 10 ML models were developed as the base classifiers, 5 each for binary and multiclass classification. The proposed WIDS was developed in the Python 3.11 programming language, using the tensorflow library and Keras API. The detailed architecture of the individual models is shown in Tables 7–11. In the case of a parameter not being specified, the default value provided by the Keras API is used. Other than the loss function used, the number of neurons, and the activation function of the output layer, binary and multiclass models are the same.

Table 7. CNN-LSTM model architecture.

Conv1D: filters = 32, kernel_size = 3, activation = ReLU, padding = same
Dropout: rate = 0.2
LSTM: units = 32, return_sequences = true
Dropout: rate = 0.2
Dense: units = 16, activation = ReLU
Dropout: rate = 0.2
Dense
units = 1, activation = sigmoid (binary classification)
units = 3, activation = softmax (multiclass classification)
Optimizer: Adam, learning_rate = 0.0001
Loss: Binary Focal Crossentropy (binary classification)
Categorical Focal Crossentropy (multiclass classification), gamma = 2.0

Table 8. CNN-GRU model architecture.

Conv1D: filters = 24, kernel_size = 3, activation = ReLU, padding = same
Dropout: rate = 0.25
GRU: units = 16, return_sequences = true
Dropout: rate = 0.25
Dense: units = 8, activation = ReLU
Dropout: rate = 0.25
Dense
units = 1, activation = sigmoid (binary classification)
units = 3, activation = softmax (multiclass classification)
Optimizer: Adam, learning_rate = 0.0001
Loss: Binary Focal Crossentropy (binary classification)
Categorical Focal Crossentropy (multiclass classification), gamma = 2.0

Table 9. Time-domain CNN model architecture.

Conv1D: filters = 16, kernel_size = 3, activation = ReLU, padding = same, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.4
Conv1D: filters = 16, kernel_size = 3, activation = ReLU, padding = same, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.4
Conv1D: filters = 16, kernel_size = 3, activation = ReLU, padding = same, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.4
Dense units = 1, activation = sigmoid (binary classification) units = 3, activation = softmax (multiclass classification) kernel_regularizer = l2, bias_regularizer = l2
Optimizer: Adam, learning_rate = 0.0001 Loss: Binary Focal Crossentropy (binary classification) Categorical Focal Crossentropy (multiclass classification), gamma = 2.0

Table 10. DNN model architecture.

Dense: units = 48, activation = ReLU, kernel_regularizer = l2
BatchNormalization
Dropout: rate = 0.2
Dense: units = 32, activation = ReLU, kernel_regularizer = l2
BatchNormalization
Dropout: rate = 0.2
Dense: units = 16, activation = ReLU, kernel_regularizer = l2
BatchNormalization
Dropout: rate = 0.2
Dense units = 1, activation = sigmoid (binary classification) units = 3, activation = softmax (multiclass classification) kernel_regularizer = l2
Optimizer: Adam, learning_rate = 0.00001 Loss: Binary Crossentropy (binary classification) Categorical Crossentropy (multiclass classification)

Two of the implemented architectures are RNN models, namely the CNN-LSTM and CNN-GRU. They both use a one-dimensional convolutional layer for feature extraction. The output of the convolutional layer is fed into an RNN-LSTM and GRU, respectively, allowing for the recognition of temporal patterns. The next model architecture is the time-domain CNN, made up of three convolutional layers, operating across multiple samples. The two remaining models are DNN and 1D-CNN, both of which analyze individual frames, lacking the ability to recognize time relationships between them. The

selected model architectures are flexible in that they may take a sequence of frames of any length as their input, allowing for easy construction of the stacked model and interoperability. They are also some of the most common and best performing ones used in state-of-the-art solutions.

Table 11. 1D-CNN model architecture.

Reshape: shape = (39, 1)
Conv1D: filters = 24, kernel_size = 1, activation = ReLU, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.3
Conv1D: filters = 12, kernel_size = 1, activation = ReLU, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.3
Flatten
Dense: units = 8, activation = ReLU, kernel_regularizer = l2, bias_regularizer = l2
BatchNormalization
Dropout: rate = 0.3
Dense units = 1, activation = sigmoid (binary classification) units = 3, activation = softmax (multiclass classification) kernel_regularizer = l2, bias_regularizer = l2
Optimizer: Adam, learning_rate = 0.00001 Loss: Binary Crossentropy (binary classification) Categorical Crossentropy (multiclass classification)

The base classifiers were evaluated on the test dataset and their performance is shown in Table 12 for binary and in Table 13 for multiclass classification. As can be seen, CNN-LSTM was the best-performing model, achieving 99.792% accuracy for binary classification and 99.706% accuracy for multiclass classification. The other RNN model, CNN-GRU, also performed well, marginally worse than CNN-LSTM in all metrics aside from higher precision. The time-domain CNN model did well in binary classification. However, it underperformed in multiclass classification. Across both classification tasks, it managed to maintain high accuracy but suffered from low recall. The DNN and 1D-CNN models were characterized by high recall and low precision, as they were trained on a balanced dataset, resulting in an overall smaller number of normal samples and the models being less biased towards the normal class.

The results of the base classifier evaluation indicate that the RNN-based models achieved the strongest overall performance among the evaluated classifiers. This observation suggests that temporal information in network traffic may provide useful information for WIDS, although further evaluation across different datasets and network environments is required to assess the generality of this observation. The classifiers operating on individual samples exhibited substantially lower precision, potentially resulting in a higher number of false positives in an IDS.

For binary and multiclass classification, the base classifiers were ranked according to their out-of-fold validation F1 scores obtained during the K-fold cross-validation procedure. The three highest-ranked and three lowest-ranked classifiers were subsequently used to build stacked models using the stacking algorithms described in Section 4.2.

The independent test dataset was employed exclusively for the final evaluation of the completed models.

Table 12. Binary models performance.

Model	Accuracy	Precision	Recall	F1 Score
CNN-LSTM	99.792%	98.872%	99.447%	99.159%
Time-Domain CNN	99.611%	99.595%	97.245%	98.406%
CNN-GRU	99.552%	99.160%	97.191%	98.166%
DNN	99.063%	93.055%	99.863%	96.339%
1D-CNN	98.701%	91.224%	98.993%	94.950%

Table 13. Multiclass models performance.

Model	Accuracy	Precision	Recall	F1 Score
CNN-LSTM	99.706%	98.848%	99.660%	99.247%
CNN-GRU	99.661%	98.852%	99.410%	99.127%
DNN	99.057%	96.165%	99.537%	97.743%
Time-Domain CNN	99.096%	99.560%	95.599%	97.488%
1D-CNN	98.975%	96.000%	99.222%	97.516%

5.6. Binary Case

For each triplet of base classifiers, four stacked models were constructed and evaluated. These are: arithmetic mean, F1 score weighted mean, logistic regression and MLP.

The performance of each stacked classifier is compared to the best of its base classifiers, which is the one with the highest F1 score. This is because a stacked model is more computationally demanding than its base classifiers. Therefore, to justify such an approach, a significant improvement in terms of performance is expected.

For the binary case, the three best-performing models were CNN-LSTM, time-domain CNN and CNN-GRU. The results of the evaluation are presented in Table 14. Of all the stacked models tested, these were the most successful ones. An increase in accuracy, precision and F1 score was seen with all of the stacking algorithms tested. In addition, with the LR and MLP meta-classifiers the model achieved higher recall than the CNN-LSTM model.

While the improvement in precision for arithmetic and weighted averaging of model outputs can be explained by the high precision of the time-domain CNN and CNN-GRU base classifiers, the increase in recall with LR and MLP-stacked models is a particularly satisfying result because the achieved metric is higher than that of any individual base classifier. The MLP was the best-stacking algorithm, providing 0.348% increase in F1 score and 0.086% increase in accuracy. The metric with the highest increase of 0.606% is precision, as can be seen in the confusion matrices of the CNN-LSTM in Figure 7 and the stacked model in Figure 8. The number of false positives more than halved, from 1120 to 516. It should be noted that other stacking algorithms had an even better increase in precision. However, they had less improvement in terms of other metrics. Hence, the MLP was deemed the best one, as it had the highest accuracy and F1 score.

Table 14. Performance of stacking algorithms for the best 3 binary models.

Base classifiers				
Model	Accuracy	Precision	Recall	F1 Score
CNN-LSTM	99.792%	98.872%	99.447%	99.159%
Time-Domain CNN	99.611%	99.595%	97.245%	98.406%
CNN-GRU	99.552%	99.160%	97.191%	98.166%
Meta-classifiers				
	Accuracy	Precision	Recall	F1 Score
Arithmetic mean	99.868%	99.730%	99.201%	99.465%
Difference from CNN-LSTM	0.076%	0.858%	−0.246%	0.309%
F1 score weighted mean	99.875%	99.730%	99.258%	99.494%
Difference from CNN-LSTM	0.083%	0.858%	−0.189%	0.333%
Logistic regression	99.876%	99.510%	99.485%	99.497%
Difference from CNN-LSTM	0.084%	0.638%	0.038%	0.338%
MLP	99.878%	99.478%	99.535%	99.507%
Difference from CNN-LSTM	0.086%	0.606%	0.088%	0.348%

The results for the next set of stacked models were less promising. Nevertheless, model stacking proved effective when combined with LR and MLP (see Table 15). The CNN-GRU model achieved a high precision of 99.160%. In contrast, the other two base classifiers, DNN and 1D-CNN, obtained lower precision scores of 93.055% and 91.224%, respectively. However, they performed significantly better in terms of recall, with the binary DNN achieving the highest recall among all models developed in the experiment at 99.863%. Despite this strong recall, their relatively low precision resulted in poorer overall performance, as reflected in lower accuracy and F1 scores.

The stacked models with arithmetic and the F1 score weighted mean stacking algorithm achieved metrics very similar to those of the DNN base classifier. This resulted in an increase in recall compared to the best of the base classifiers, CNN-GRU. However, the even greater decrease in precision yielded lower accuracy and an F1 score. In contrast, the use of LR and MLP as meta-classifiers proved more effective. Although the stacked models still exhibited lower precision than CNN-GRU, this reduction was offset by a more substantial gain in recall, leading to improvements in both accuracy and F1 score. The smallest precision loss was observed with the MLP meta-classifier (0.883%), which was considerably lower than the corresponding recall gain of 1.852%. Overall, the MLP-based stacked model achieved the best performance, attaining 99.668% accuracy and a 98.659% F1 score. This demonstrates its ability to effectively compensate for the weaknesses of the base classifiers and enhance overall performance. The confusion matrices of the CNN-GRU model (left) and the stacked model with the MLP meta-classifier are shown in Figures 9 and 10, where a higher number of false positives and fewer false negatives can be observed in the stacked model.

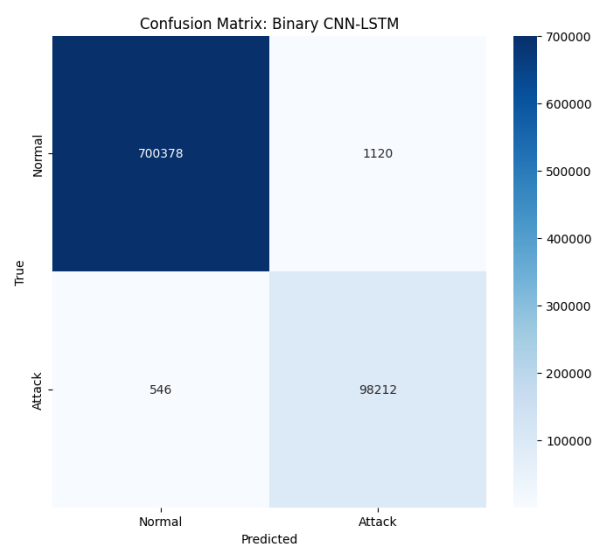


Figure 7. Confusion matrix of binary CNN-LSTM model.

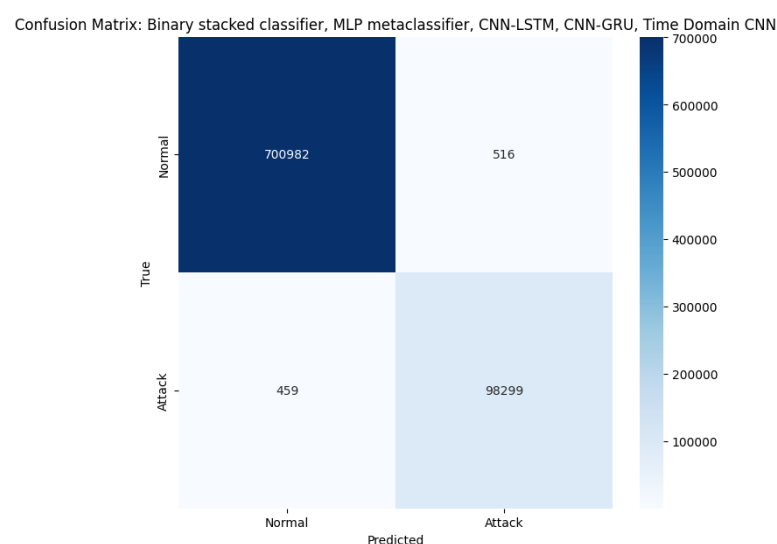


Figure 8. Confusion matrix of stacked model with MLP meta-classifier with 3 best base classifier.

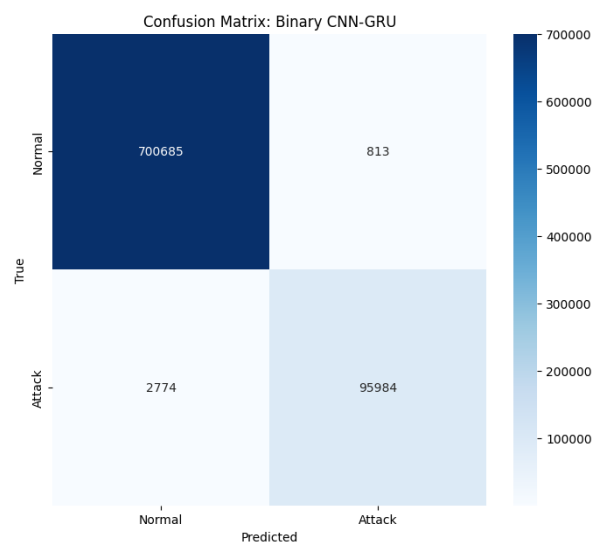


Figure 9. Confusion matrix of binary CNN-GRU model.

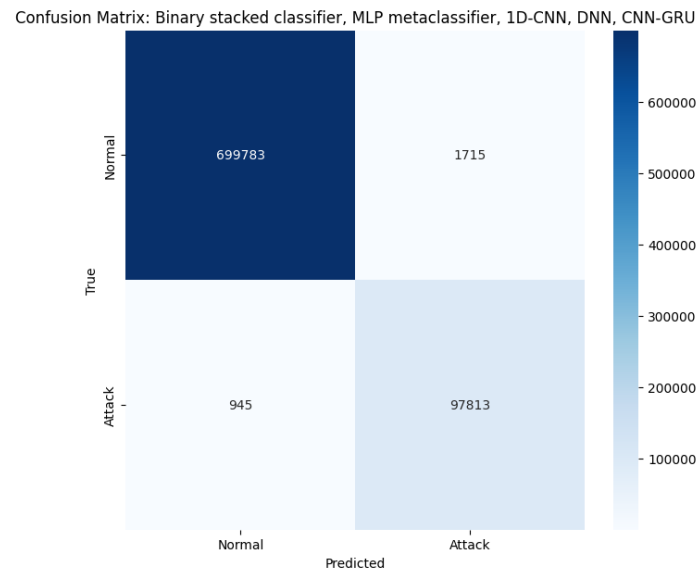


Figure 10. Confusion matrix of stacked model with MLP meta-classifier with 3 worst base classifier.

Table 15. Performance of stacking algorithms for the worst 3 binary models.

Base classifiers				
Model	Accuracy	Precision	Recall	F1 Score
CNN-GRU	99.552%	99.160%	97.191%	98.166%
DNN	99.063%	93.055%	99.863%	96.339%
1D-CNN	98.701%	91.224%	98.993%	94.950%
Meta-classifiers				
	Accuracy	Precision	Recall	F1 Score
Arithmetic mean	99.056%	93.005%	99.863%	96.312%
Difference from CNN-GRU	−0.496%	−6.155%	2.672%	−1.854%
F1 score weighted mean	99.059%	93.026%	99.862%	96.323%
Difference from CNN-GRU	−0.492%	−6.134%	2.671%	−1.843%
Logistic regression	99.658%	97.960%	99.297%	98.624%
Difference from CNN-GRU	0.106%	−1.200%	2.106%	0.458%
MLP	99.668%	98.277%	99.043%	98.659%
Difference from CNN-GRU	0.116%	−0.883%	1.852%	0.493%

5.7. Multiclass Case

For multiclass classification, the three best models were CNN-LSTM, CNN-GRU, and DNN (see Table 16). They all had very high recall, above 99%. Unfortunately, they performed worse in terms of precision, especially the DNN, which is consistent with the results for binary classification. The best of the three base classifiers was CNN-LSTM, which reliably proved to be the best-performing model architecture for both classification tasks.

In the case of arithmetic and weighted means, the metrics were almost identical to those of the DNN. The results are similar to those of the binary stacked classifier with the three worst-performing base classifiers. This may indicate that these methods are sensitive to the relatively worse-performing base classifiers, which may hinder the quality of the stacked model. As for regression and MLP, they achieved a significant increase in precision

with minimally worse recall. The precision gain was 0.647% for the regression and 0.764% for the MLP meta-classifier, which greatly offsets the loss of recall of 0.018% and 0.040%, respectively, compared to CNN-LSTM, resulting in an increase in both accuracy and F1 score. The MLP meta-classification performed slightly better than regression. The confusion matrices of CNN-LSTM multiclass and the stacked model with the MLP meta-classifier are shown in Figures 11 and 12. As can be seen, both models are very accurate in detecting Impersonation attacks, with very few mistakes in terms of both false positives and false negatives. The mistakes that confuse the two attack classes are also very rare. The most common errors are falsely detected. Flooding attacks, the number of which has been greatly reduced by the stacked classifier.

The other stacked model utilizes DNN, time-domain CNN and 1D-CNN as its base classifiers (see Table 17). As was the case for binary models, the DNN and 1D-CNN are characterized by high recall and low precision. The time-domain CNN had the highest precision of all multiclass base classifiers. However, due to low recall, it ranked last in terms of F1 score, despite achieving slightly higher accuracy than the DNN and 1D-CNN.

The arithmetic and weighted mean stacking methods did not differ much from the performance of DNN, with very marginal changes to precision and recall. Regression and MLP meta-classifiers resulted in a substantial increase in accuracy and precision at the cost of lower recall. In particular, the MLP meta classifier had the highest precision of all multiclass models, even higher than time-domain CNN at 99.820%, a 3.655% increase from DNN; however, it came with a 2.717% decrease in recall. The softmax regression meta-classifier was more balanced in that the gain in precision was lower at 2.320% and the loss of recall was -1.182% . This resulted in better overall performance with an improvement of 0.318% in accuracy and an improvement of 0.677% in the F1 score. The confusion matrices of the DNN and softmax regression stacked classifier can be seen in Figures 13 and 14. As was the case in the previous scenario, both the DNN-base classifier and the stacked model were very accurate in identifying the Impersonation attacks, less so for the flooding class. In particular, the DNN had 100% recall in the Flooding class, which resulted in a large number of false positives. The meta-classifier managed to compensate for this, significantly reducing the number of false positives, with the drawback of lower recall.

Table 16. Performance of stacking algorithms for the best 3 multiclass models.

Base classifiers				
Model	Accuracy	Precision	Recall	F1 Score
CNN-LSTM	99.706%	98.848%	99.660%	99.247%
CNN-GRU	99.661%	98.852%	99.410%	99.127%
DNN	99.057%	96.165%	99.537%	97.743%
Meta-classifiers				
	Accuracy	Precision	Recall	F1 Score
Arithmetic mean	99.068%	96.248%	99.537%	97.784%
Difference from CNN-LSTM	-0.638%	-2.600%	-0.123%	-1.463%
F1 score weighted mean	99.070%	96.269%	99.526%	97.789%
Difference from CNN-LSTM	-0.636%	-2.579%	-0.134%	-1.458%
Softmax regression	99.836%	99.495%	99.642%	99.568%
Difference from CNN-LSTM	0.130%	0.647%	-0.018%	0.321%
MLP	99.858%	99.612%	99.620%	99.616%
Difference from CNN-LSTM	0.152%	0.764%	-0.040%	0.369%

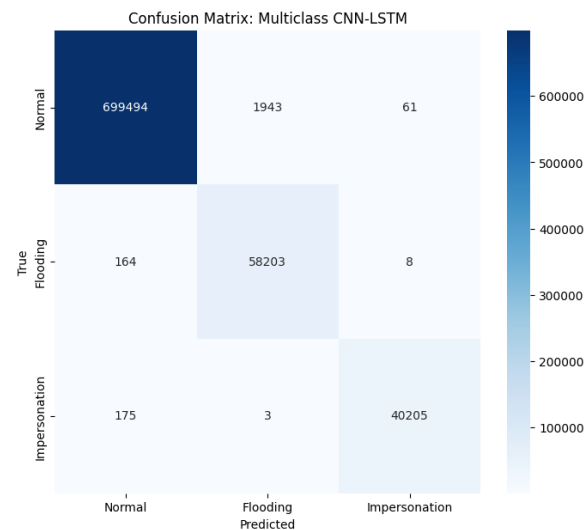


Figure 11. Confusion matrix of multiclass CNN-LSTM model.

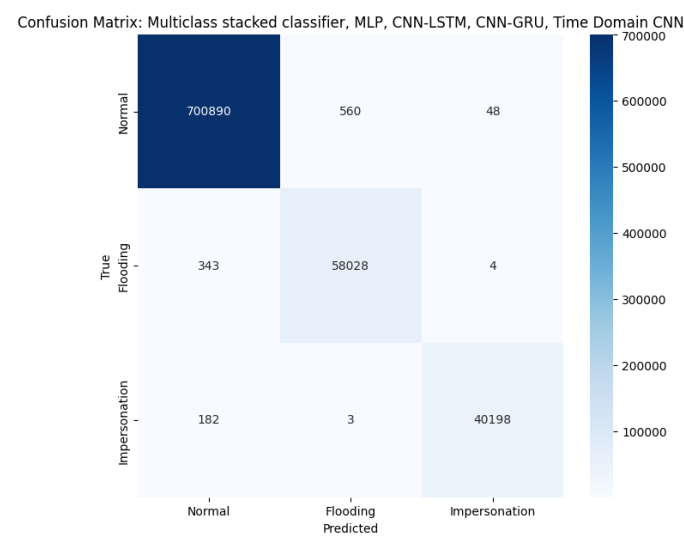


Figure 12. Confusion matrix of multiclass stacked model with MLP meta-classifier with 3 best base classifiers.

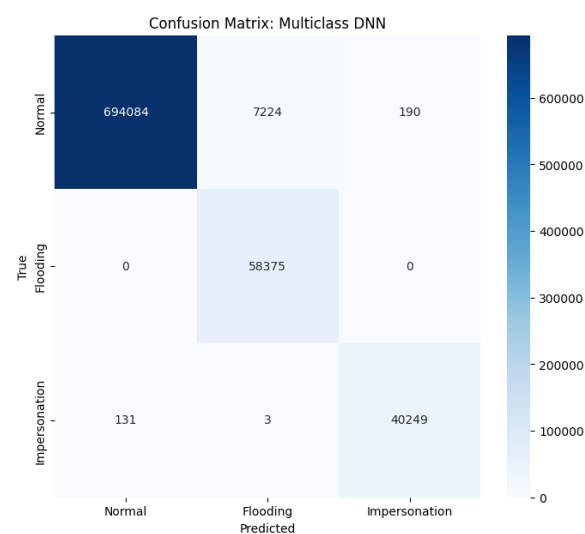
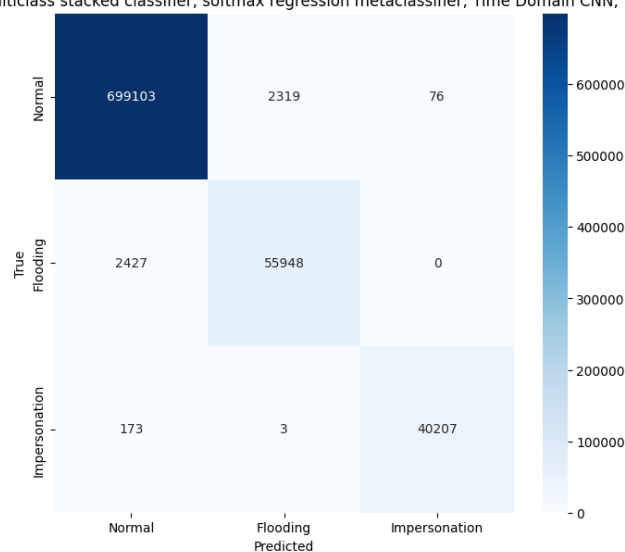


Figure 13. Confusion matrix of multiclass DNN model.

Table 17. Performance of stacking algorithms for the worst 3 multiclass models.

Base classifiers				
Model	Accuracy	Precision	Recall	F1 Score
DNN	99.057%	96.165%	99.537%	97.743%
Time-Domain CNN	99.096%	99.560%	95.599%	97.488%
1D-CNN	98.975%	96.000%	99.222%	97.516%
Meta-classifiers				
	Accuracy	Precision	Recall	F1 Score
Arithmetic mean	99.057%	96.183%	99.526%	97.748%
Difference from DNN	0.000%	0.018%	−0.011%	0.005%
F1 weighted mean	99.058%	96.183%	99.530%	97.749%
Difference from DNN	0.001%	0.018%	−0.007%	0.006%
Softmax regression	99.375%	98.485%	98.355%	98.420%
Difference from DNN	0.318%	2.320%	−1.182%	0.677%
MLP	99.305%	99.820%	96.820%	98.178%
Difference from DNN	0.248%	3.655%	−2.717%	0.435%

Confusion Matrix: Multiclass stacked classifier, softmax regression metaclassifier, Time Domain CNN, 1D-CNN, DNN

**Figure 14.** Confusion matrix of stacked model with softmax regression meta-classifier with 3 worst base classifiers.

5.8. Performance

The computational efficiency of the proposed WIDS system is a pressing issue; therefore, the inference time of the base classifiers, as well as the two best-performing stacked models from Section 5.7, was measured. The experiment was carried out on a PC equipped with an Intel i5-12600KF CPU and an Nvidia RTX 3060 Ti GPU, running the Fedora 42 operating system. The measured inference times are presented in Table 18; the model was inferring on batches of four sequences with 64 frames, each with 39 features. Using a small batch size is the most accurate way of measuring a real-time scenario, as the network traffic captured by a WIDS would come in small batches. The results demonstrate that the overhead introduced by model stacking is negligible in terms of latency, as the difference between the stacked model and its slowest base classifier is sub-millisecond. Even for the

slowest of the models tested, the throughput is over 100,000 frames per second, significantly more than would be present in any real-world network.

Table 18. Model inference time.

Model Inference Time				
Model	Inference Time ms	Standard Deviation	Frames per Second	Number of Parameters
CNN-LSTM	1.188	0.063	215,480	33,411
CNN-GRU	0.900	0.047	284,526	28,131
Time-Domain CNN	0.853	0.151	300,207	5995
DNN	0.595	0.048	430,397	4451
1D CNN	1.152	0.118	222,170	4543
Stacked CNN-LSTM, CNN-GRU, DNN, MLP meta-classifier	1.688	0.228	151,686	66,068
Stacked DNN, TD CNN, 1D CNN, regression meta-classifier	1.974	0.386	129,710	15,019

6. Conclusions and Future Work

In the paper, it was explored how implementing a stacking algorithm can enhance the performance of a WIDS. The authors proposed a WIDS based on model stacking. Various stacking algorithms and ML architectures were also explored and evaluated. It was shown how model stacking can be implemented for intrusion detection, particularly how different ML algorithms, such as RNN and CNN, can be integrated to enhance the security of wireless networks. The proposed methodology was proven to enhance threat detection in wireless networks, especially to increase the precision of an IDS, which is a pressing issue for such systems, as they suffer from a high false alert rate.

The performance of the proposed solution is on par with existing state-of-the-art solutions, making it suitable for WIDS applications. The experimental evaluation demonstrates that heterogeneous stacking constitutes an effective approach for Wi-Fi intrusion detection when appropriate aggregation strategies are employed. The obtained results also indicate that the choice of the aggregation method significantly influences the final detection performance, and not all stacking configurations provide identical benefits across all evaluation metrics. For each of the scenarios tested, model stacking was shown to substantially enhance classifier performance, leading to higher accuracy and F1 score, and more often than not, precision. Improvements in these metrics were, in some cases, accompanied by a reduction in recall. The improvements, while numerically small (i.e., the 0.15% accuracy increase in the multiclass scenario), come with a significant decrease in wrong classifications. Nevertheless, the overall enhancement in system performance was substantial enough to justify the proposed solution. It should also be noted that the AWID3 dataset is already a highly optimized benchmark on which performance is approaching saturation, making further gains increasingly difficult to achieve. At this level of performance, even modest improvements correspond to a reduction in the number of misclassified frames, which can be significant in real wireless networks processing millions of frames. Moreover, the proposed approach achieves this improvement while relying only on features available from the first and second layers of the IEEE 802.11 protocol stack, making it suitable for deployment without requiring higher-layer traffic inspection.

Although the proposed framework achieves excellent performance on the AWID3 benchmark, the reported results should not be interpreted as evidence of universal generalization to all IEEE 802.11 deployments. Like many publicly available intrusion detection

datasets, AWID3 represents a limited number of network environments and attack scenarios, which may introduce dataset-specific bias. To reduce this risk, the proposed feature set was intentionally restricted to protocol-level IEEE 802.11 MAC-layer attributes expected to remain consistent across different WLAN deployments. Nevertheless, a comprehensive evaluation of generalization would require cross-capture validation, leave-one-scenario-out experiments, or testing on independent datasets collected in different network environments. Such an evaluation is left for future work. Future directions of research may also include the incorporation of other ML architectures, such as Autoencoders and Decision Trees, to compose a more heterogeneous set of base classifiers that might excel in detecting different types of attacks. Another important direction for future research is the implementation and evaluation of the proposed framework in real-world WLAN environments. In particular, future work will investigate adaptive stacking strategies that dynamically adjust the number of active base classifiers according to changing traffic intensity and available computational resources. A dedicated evaluation on embedded and resource-constrained hardware platforms, including measurements of inference latency, memory footprint, and energy consumption, also constitutes an important direction for future work.

Author Contributions: Conceptualization, M.N.; methodology, F.W. and M.N.; software, F.W.; validation, F.W. and M.N.; formal analysis, F.W. and M.N.; investigation, F.W. and M.N.; writing—original draft preparation, F.W. and M.N.; writing—review and editing, M.N.; visualization, F.W.; supervision, M.N.; project administration, M.N.; funding acquisition, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Research Institute, grant number POIR.04.02.00-00-D008/20-01, the “National Laboratory for Advanced 5G Research” (acronym PL-5G), as part of Measure 4.2, ‘Development of modern research infrastructure of the science sector’ 2014–2020, financed by the European Regional Development Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
ANN	Artificial Neural Network
AP	Access Point
AWID3	Aegean Wi-Fi Intrusion Dataset 3
BiLSTM	Bidirectional LSTM
CCK	Complementary Code Keying
CCMP	CCM mode Protocol
CNN	Convolutional Neural Networks
CSMA/CA	Carrier Sensing Multiple Access Collision Avoidance
CTS	Clear-to-Send
CW	Contention Window
DBN	Deep Belief Network
DL	Deep Learning
DT	Decision Tree

DoS	Denial of Service
DS	Distribution System
ESS	Extended Service Set
FCN	Fully Convolutional Network
GAF	Gramian Angular Field
GBT	Gradient Boosting Trees
GRU	Gated Recurrent Unit
HRC	Hybrid Regression-Classification
IDS	Intrusion Detection System
kNN	k Nearest Neighbors
LLMs	Large Language Models
LSTM	Long Short-Term Memory
LR	Logistic Regression
MitM	Man-in-the-Middle
ML	Machine Learning
MLP	Multilayer Perceptron
NBC	Naive-Bayes Classifier
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
PMF	Protected Management Frames
PSK	Pre-Shared Key
RBM	Restricted Boltzmann Machine
RF	Random Forest
RNN	Recurrent Neural Networks
RTS	Request-to-Send
SAE	Simultaneous Authentication of Equals
SNR	Signal-to-Noise Ratio
SSID	Service Set Identifier
SVM	Support Vector Machines
TK	Transmission Key
TKIP	Temporal Key Integrity Protocol
WEP	Wired Equivalent Privacy
WIDS	Wireless Intrusion Detection System
WLANs	Wireless Local Area Networks
WPA	Wi-Fi Protected Access
ViT	Vision Transformer
XGB	Extreme Gradient Boosting

References

1. *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*; IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standards Association: Piscataway, NJ, USA, 1998; pp. 1–512.
2. *IEEE Std 802.11i-2004*; IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Standards Association: Piscataway, NJ, USA, 2004; pp. 1–190. [[CrossRef](#)]
3. *IEEE Std 802.11-2020/Cor 1-2022 (Corrigendum to IEEE Std 802.11-2020 as Amended by IEEE Std 802.11ax-2021, IEEE Std 802.11ay-2021, and IEEE Std 802.11ba-2021)*; IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Corrigendum 1–Correct IEEE 802.11ay Assignment of Protected Announce Support Bit. IEEE Standards Association: Piscataway, NJ, USA, 2022; pp. 1–18. [[CrossRef](#)]

4. IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as Amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008); IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Protected Management Frames. IEEE Standards Association: Piscataway, NJ, USA, 2009; pp. 1–111. [\[CrossRef\]](#)
5. Larbig, P. MDK3. 2018. Available online: <https://github.com/aircrack-ng/mdk3> (accessed on 23 February 2026).
6. d’Otreppe, T.; The Aircrack-ng team. Aircrack-ng, a Complete Suite of Tools to Assess WiFi Network Security, Version 1.7. 2022. Available online: <https://www.aircrack-ng.org> (accessed on 23 February 2026).
7. Vanhoef, M.; Piessens, F. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS’17*; ACM: New York, NY, USA, 2017; pp. 1313–1328. [\[CrossRef\]](#)
8. Cermak, M.; Lipovsky, R.; Svorencik, S. KR00K–CVE-2019-15126; Technical Report; ESET: North Sydney, Australia, 2020.
9. Stryczek, S.; Natkaniec, M. Internet Threat Detection in Smart Grids Based on Network Traffic Analysis Using LSTM, IF, and SVM. *Energies* **2023**, *16*, 329. [\[CrossRef\]](#)
10. Vaca, F.D.; Niyaz, Q. An Ensemble Learning Based Wi-Fi Network Intrusion Detection System (WNIDS). In *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5. [\[CrossRef\]](#)
11. Munna, M.M.I.; Rahman, M.M.; Frnda, J.; Anwar, M.S.; Kutlimuratov, A. Elevating intrusion detection and security fortification in intelligent networks through cutting-edge machine learning paradigms. *Sci. Rep.* **2025**, *15*, 39989. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Dashtifard, N.; Mahmoud, H.; Idrissi, M.; Elmitwally, N. Enhanced Anomaly Detection in Wireless 5G Networks With Hybrid Learning Technique Using AWID3 Dataset. *Cureus J. Comput. Sci.* **2025**, *2*, es44389-025-05486-0. [\[CrossRef\]](#)
13. Uszko, K.; Kasprzyk, M.; Natkaniec, M.; Cholda, P. Rule-Based System with Machine Learning Support for Detecting Anomalies in 5G WLANs. *Electronics* **2023**, *12*, 2355. [\[CrossRef\]](#)
14. Saini, R.; Halder, D.; Baswade, A.M. RIDS: Real-time Intrusion Detection System for WPA3 enabled Enterprise Networks. In *Proceedings of the GLOBECOM 2022–2022 IEEE Global Communications Conference*; IEEE: Piscataway, NJ, USA, 2022; pp. 43–48. [\[CrossRef\]](#)
15. Warhekar, A.S.; Kshirsagar, D.D. A Wireless Intrusion Detection System using Feature Selection with Random Forest. In *Proceedings of the 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–4. [\[CrossRef\]](#)
16. El Bizri, M.; El-Hajj, A.M.; Haidar, A.M. Towards Safer Wi-Fi Networks: Leveraging Neural Networks for Intrusion Detection. In *Proceedings of the 2023 IEEE 4th International Multidisciplinary Conference on Engineering Technology (IMCET)*; IEEE: Piscataway, NJ, USA, 2023; pp. 273–277. [\[CrossRef\]](#)
17. Halbouni, A.H.; Gunawan, T.S.; Halbouni, M.; Assaig, F.A.A.; Effendi, M.R.; Ismail, N. CNN-IDS: Convolutional Neural Network for Network Intrusion Detection System. In *Proceedings of the 2022 8th International Conference on Wireless and Telematics (ICWT)*; IEEE: Piscataway, NJ, USA, 2022; pp. 1–4. [\[CrossRef\]](#)
18. Jouhari, M.; Guizani, M. Lightweight CNN-BiLSTM based Intrusion Detection Systems for Resource-Constrained IoT Devices. In *Proceedings of the 2024 International Wireless Communications and Mobile Computing (IWCMC)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1558–1563. [\[CrossRef\]](#)
19. Natkaniec, M.; Bednarz, M. Wireless Local Area Networks Threat Detection Using 1D-CNN. *Sensors* **2023**, *23*, 5507. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Ahmad, R.S.; Ahmad, R.; Niyaz, Q. Lightweight CNN-Based Wi-Fi Intrusion Detection Using 2D Traffic Representations. *arXiv* **2025**. [\[CrossRef\]](#)
21. Zeng, W.; Han, D.; Cui, M.; Wu, Z.; Han, B.; Zhou, H. IFLV: Wireless network intrusion detection model integrating FCN, LSTM, and ViT. In *Proceedings of the 2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*; IEEE: Piscataway, NJ, USA, 2023; pp. 470–475. [\[CrossRef\]](#)
22. Şolpan, Ş.; Gündüz, H.; Küçük, K. Wi-Fi Network Intrusion Detection: Enhanced with Feature Extraction and Machine Learning Algorithms. In *Proceedings of the 2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–7. [\[CrossRef\]](#)
23. Nguyen Dang, K.D.; Fazio, P.; Voznak, M. A Novel Deep Learning Framework for Intrusion Detection Systems in Wireless Network. *Future Internet* **2024**, *16*, 264. [\[CrossRef\]](#)
24. Zhao, R.; Mu, Y.; Zou, L.; Wen, X. A Hybrid Intrusion Detection System Based on Feature Selection and Weighted Stacking Classifier. *IEEE Access* **2022**, *10*, 71414–71426. [\[CrossRef\]](#)
25. Lee, S.J.; Yoo, P.D.; Asyhari, A.T.; Jhi, Y.; Chermak, L.; Yeun, C.Y.; Taha, K. IMPACT: Impersonation Attack Detection via Edge Computing Using Deep Autoencoder and Feature Abstraction. *IEEE Access* **2020**, *8*, 65520–65529. [\[CrossRef\]](#)

26. Ravi, N.; Rani, P.V.; Shalinie, S.M. Secure Deep Neural (SeDeN) Framework for 5G Wireless Networks. In *Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [[CrossRef](#)]
27. Mathappan, N.; Suganya, E.; Sountharajan, S.; Madhavan, P.; Bavirisetti, D. FEDDBN-IDS: Federated Deep Belief Network-based Wireless Network Intrusion Detection System. *EURASIP J. Inf. Secur.* **2024**, *2024*, 8. [[CrossRef](#)]
28. Yonbawi, S.; Afzal, A.; Yasir, M.; Rizwan, M.; Kryvinska, N. Transferability Evaluation in Wi-Fi Intrusion Detection Systems Through Machine Learning and Deep Learning Approaches. *IEEE Access* **2025**, *13*, 11248–11264. [[CrossRef](#)]
29. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 184–208. [[CrossRef](#)]
30. Chatzoglou, E.; Kambourakis, G.; Kolias, C. Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset. *IEEE Access* **2021**, *9*, 34188–34205. [[CrossRef](#)]
31. Chatzoglou, E.; Kambourakis, G.; Kolias, C.; Smiliotopoulos, C. Pick Quality Over Quantity: Expert Feature Selection and Data Preprocessing for 802.11 Intrusion Detection Systems. *IEEE Access* **2022**, *10*, 64761–64784. [[CrossRef](#)]
32. Ledwoń, A.; Natkaniec, M. Machine Learning for Wi-Fi Intrusion Detection: A Comparative Study of Accuracy, Explainability, and Adversarial Robustness. *IEEE Access* **2026**, *14*, 7582–7599. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.