electronics

MDPI

Article

Machine Learning-Based Multilabel Classification for Web Application Firewalls: A Comparative Study

Cristian Chindrus † and Constantin-Florin Caruntu *,†

Department of Automatic Control and Applied Informatics, "Gheorghe Asachi" Technical University of Iasi, 700050 Iasi, Romania; cristian.chindrus@student.tuiasi.ro

* Correspondence: caruntuc@ac.tuiasi.ro

† These authors contributed equally to this work.

Abstract

The increasing complexity of web-based attacks requires the development of more effective Web Application Firewall (WAF) systems. In this study, we extend previous work by evaluating and comparing the performance of seven machine learning models for multilabel classification of web traffic, using the ECML/PKDD 2007 dataset. This dataset contains eight classes: seven representing different types of attacks and one representing normal traffic. Building on prior experiments that analyzed Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) models, we incorporate four additional models frequently cited in the related literature: Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting (GB), and Feedforward Neural Networks (NN). Each model is trained and evaluated under consistent preprocessing and validation protocols. We analyze their performance using key metrics such as accuracy, precision, recall, F1-score, and training time. The results provide insights into the suitability of each method for WAF classification tasks, with implications for real-time intrusion detection systems and security automation. This study represents the first unified multilabel evaluation of classical and deep learning approaches on the ECML/PKDD 2007 dataset, offering guidance for practical WAF deployment.

Keywords: web application firewall; machine learning; multilabel classification; recurrent neural network; gated recurrent unit; long short-term memory; random forest; support vector machine; gradient boosting; neural networks

check for updates

Academic Editor: Wajeb Gharibi

Received: 26 September 2025
Revised: 22 October 2025
Accepted: 22 October 2025
Published: 25 October 2025

Citation: Chindrus, C.; Caruntu, C.-F. Machine Learning-Based Multilabel Classification for Web Application Firewalls: A Comparative Study. *Electronics* **2025**, *14*, 4172. https://doi.org/10.3390/electronics14214172

Copyright: © 2025 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) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Web applications are increasingly becoming prime targets for cyberattacks due to their ubiquity and the critical data they handle. As the complexity and volume of these attacks continue to grow, traditional rule-based Web Application Firewalls (WAFs) are proving insufficient. These conventional systems rely on predefined signatures or manually curated rules, making them less effective against novel or obfuscated threats. Machine learning (ML) techniques offer a promising alternative, enabling systems to learn from past traffic patterns and detect both known and previously unseen attacks [1].

Over the past decade, researchers have explored a variety of ML models to improve the accuracy and adaptability of intrusion detection systems (IDS) and WAFs [2]. Among the most studied approaches are deep learning models like Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM) networks, known for their ability to model sequential and temporal dependencies in data. In our

previous work, we applied these three deep learning models to the ECML/PKDD 2007 dataset [3], a well-known benchmark for web traffic analysis, which includes both normal traffic and multiple types of web attacks. The study demonstrated the capability of these models to handle the complex patterns inherent in multilabel classification of web traffic. However, the comparison was limited to only deep learning approaches.

In parallel, other researchers have evaluated traditional machine learning models, such as Support Vector Machines (SVM), Random Forests (RF), Gradient Boosting (GB), and basic Feedforward Neural Networks (NN), in similar contexts. These models have shown strong performance across various network intrusion datasets, including KDD'99, NSL-KDD [4], and CIC-IDS2017 [5]. However, their effectiveness on the ECML/PKDD 2007 dataset, particularly for multilabel classification of web traffic, has not been extensively explored. These evaluations were typically performed on different datasets with differing features, traffic patterns, and attack types, making direct comparisons difficult and potentially misleading.

This paper aims to fill that gap by presenting a unified evaluation of both deep learning and classical machine learning models on the same dataset, the ECML/PKDD 2007 corpus. This dataset is particularly well-suited for multilabel classification tasks as it contains traffic labeled with eight distinct classes: seven representing different types of web attacks (such as SQL injection, Cross-Site Scripting (XSS), and others), and one representing normal traffic. By applying all seven models (RNN, GRU, LSTM, Random Forest, SVM, Gradient Boosting, and NN) to the same dataset, under consistent preprocessing and evaluation protocols, we can fairly assess their strengths, weaknesses, and overall suitability for multilabel WAF classification.

Our primary objective is to determine which machine learning approach is most appropriate for deployment in a WAF context, where accurate and efficient multilabel classification of web traffic is crucial. Each model is evaluated using standard classification metrics, including accuracy, precision, recall, F1-score, and training time. In addition to overall performance, we analyze the models' ability to detect specific attack types, highlighting whether certain algorithms are more sensitive to particular patterns or vulnerabilities in the traffic.

This paper contributes to the field in three key ways. First, it extends our previous study [6] on deep learning models by introducing a direct comparison with classical models, thereby broadening the scope of analysis. Second, it standardizes the evaluation by using a single, well-established dataset across all models, which helps eliminate inconsistencies caused by varying dataset characteristics. Third, it provides a detailed performance breakdown, including per-class evaluation and training efficiency, offering practical insights for researchers and practitioners aiming to implement ML-based WAF systems in real-world environments.

2. Related Work

This section reviews previous research focused on applying machine learning techniques to web application firewalls, with special attention to studies using the ECML/PKDD 2007 dataset. It also compares traditional and deep learning models used in similar security tasks, highlighting the gaps this paper addresses.

2.1. Machine Learning in Web Application Firewalls

Web Application Firewalls are designed to protect web applications by monitoring and filtering HTTP/S traffic, identifying and blocking potentially harmful requests. Traditional WAFs operate based on predefined rules or signatures, which limit their ability to detect new or evolving attacks. To address this limitation, many researchers have proposed

integrating machine learning techniques into WAFs to enhance their detection capabilities and reduce manual rule maintenance.

ML-based WAF systems use algorithms trained on historical web traffic data to classify incoming requests as normal or malicious [7]. Unlike rule-based systems, ML models can learn patterns and make predictions on previously unseen traffic, including obfuscated or modified attack payloads.

The machine learning models commonly used for WAF and intrusion detection tasks fall into three categories: classical models, basic neural networks, and deep learning models. Each has different strengths and requirements in terms of data, computational resources, and accuracy.

2.1.1. Classical Machine Learning Models in WAF

Classical models such as Random Forests, Support Vector Machines, and Gradient Boosting are widely used in network security tasks due to their effectiveness on structured data and relatively low computational cost. These models typically require manual feature selection and engineering but can achieve good performance with proper tuning [8].

Typical machine learning algorithms used in this study include Random Forests, which combine multiple decision trees and perform effectively on high-dimensional data; SVMs, which are well suited for classification tasks with clearly defined class boundaries; and Gradient Boosting models, such as XGBoost, which construct decision trees sequentially while iteratively minimizing classification errors to improve predictive accuracy.

These models are efficient and interpretable, making them suitable for environments where model transparency is important. However, they are less suited for capturing temporal patterns or sequential relationships in data.

2.1.2. Neural Networks in WAF

Feedforward Neural Networks (also called fully connected NNs) represent a more basic form of deep learning, where data flows in one direction through input, hidden, and output layers. They can learn nonlinear relationships between features, but do not consider the order or timing of input data [7]. As a result, they may miss temporal dependencies that are important in detecting some types of web attacks. However, their simplicity and lower computational demand compared to RNNs or LSTMs make them a useful baseline in many intrusion detection systems.

2.1.3. Deep Learning Models in WAF

Advanced deep learning models, such as Recurrent Neural Networks, Gated Recurrent Units, and Long Short-Term Memory networks, are designed to handle sequential data. These architectures are especially useful for analyzing web traffic logs, where the order of requests can be an important factor in identifying malicious behavior [9].

Common architectures of recurrent neural networks include the RNN, which is capable of modeling short-term dependencies in sequential data but often faces difficulties when handling longer sequences. The GRU represents a simplified variant of the LSTM architecture, designed to preserve comparable performance while reducing computational complexity. The LSTM network, on the other hand, excels at capturing long-term dependencies within sequences, making it particularly suitable for detecting attack patterns that unfold across multiple stages of web requests.

These models require more data and processing power but can automatically learn relevant features from raw input, reducing the need for manual engineering.

2.1.4. Applications in WAF Research

Various research studies have used different combinations of these models for WAF and intrusion detection systems [1]. However, most papers focus on either classical or deep learning models and evaluate them on different datasets, which makes comparisons difficult.

This gap motivates the current paper, which aims to evaluate and compare seven different ML models, RNN, GRU, LSTM, Random Forest, SVM, Gradient Boosting, and Feedforward NN, on the same dataset (ECML/PKDD 2007), using a consistent experimental setup.

2.2. Previous Studies Using the ECML/PKDD 2007 Dataset

The ECML/PKDD 2007 dataset, also known as the PASCAL Large Scale Learning Challenge dataset, was originally released for a challenge on discovering knowledge from web traffic logs [3]. It has since been used in various machine learning and intrusion detection studies due to its real-world nature and labeled HTTP requests, which include both normal and malicious traffic. The dataset contains a total of eight labels: seven types of attacks (e.g., SQL injection, XSS, directory traversal) and one label for normal traffic.

Unlike synthetic datasets such as KDD'99 or NSL-KDD [4], ECML/PKDD 2007 represents actual web traffic captured from a honeypot environment, making it more realistic and challenging for classification tasks. Furthermore, it supports multilabel classification, as certain requests may belong to more than one attack category.

Despite its quality and availability, relatively few published studies have used this dataset extensively, particularly in the context of multilabel classification. Most existing work focuses either on binary classification (attack vs. normal) or multiclass classification (selecting one dominant label per sample) [10]. This has left an open gap in exploring models that can predict multiple labels simultaneously, which is often required in real-world WAF scenarios where one request may exploit several vulnerabilities.

2.2.1. Limitations in Previous Use

Most previous studies on the ECML/PKDD 2007 dataset exhibit several methodological limitations. A limited variety of models has typically been explored, with most works evaluating only one or two algorithms rather than comparing both classical and deep learning approaches under uniform conditions. Furthermore, although the dataset inherently supports multilabel annotations, many researchers have simplified the problem to a multiclass format by assigning a single label to each instance. This reduction leads to a loss of information and imposes unrealistic assumptions about how web attacks occur in practice. In addition, inconsistencies in preprocessing procedures, data partitioning strategies, and evaluation metrics have made it difficult to establish a fair basis for comparing results across studies.

Another recurring limitation is the minimal attention given to hyperparameter tuning, with several works employing default parameter settings that may not yield optimal model performance. These shortcomings underscore the need for a more comprehensive and standardized evaluation framework that examines a diverse range of machine learning algorithms under the same conditions, with particular emphasis on preserving the multilabel nature of the problem. Addressing these gaps serves as the principal motivation for the present study.

2.2.2. Dataset Structure and Label Distribution

To better understand the nature of the ECML/PKDD 2007 dataset, Figure 1 shows the label distribution across the eight categories [10]. The dataset is imbalanced, with a

high volume of normal traffic and a smaller number of samples for certain attack types like "XSS" and "Command Injection".

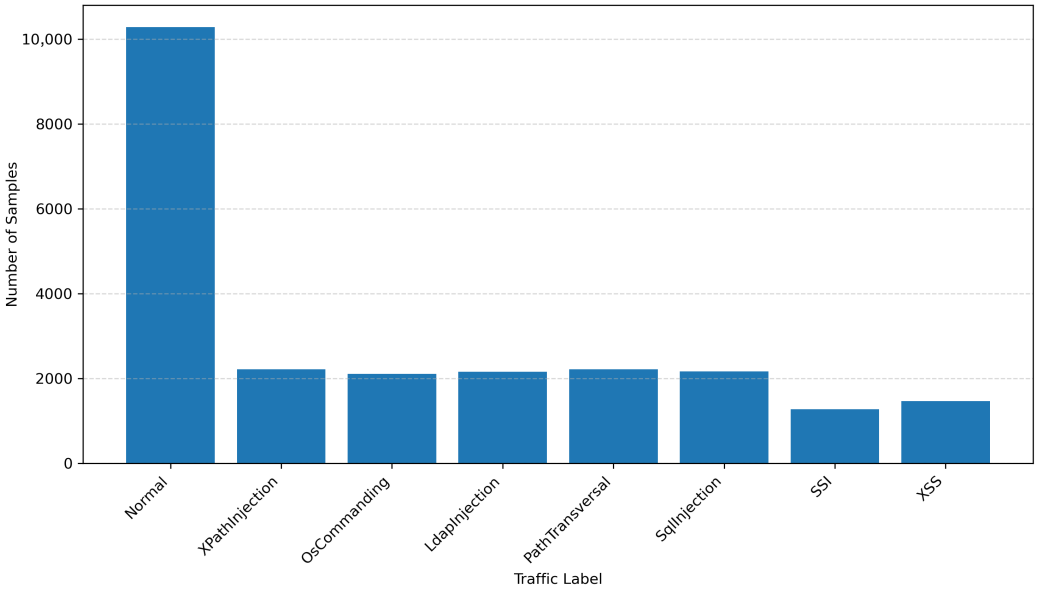


Figure 1. Distribution of Labels in the ECML/PKDD 2007 dataset.

The ECML/PKDD 2007 dataset remains one of the few publicly available resources that authentically captures the characteristics of real web traffic, encompassing both legitimate and malicious HTTP requests. It uniquely supports multilabel annotations, where a single request may be associated with multiple attack categories, an essential feature for realistic modeling of complex intrusion scenarios. Moreover, the dataset enables evaluation in a WAF, relevant context, as it preserves real-world input structures, including URLs, query parameters, and HTTP headers.

Despite these strengths, the dataset has not been extensively utilized in the literature for multilabel machine learning evaluations, particularly in comparative studies involving both classical and deep learning models. This study seeks to fill that gap by providing a structured and consistent performance comparison across seven machine learning algorithms, all tested under the same classification framework and preprocessing conditions.

2.3. Classical vs. Deep Learning Models in Attack Detection

Machine learning has become a cornerstone of modern intrusion detection systems and Web Application Firewalls, enabling automated classification of network and application-layer traffic. The models used in this field typically fall into two broad categories: classical machine learning models and deep learning models. Each category has distinct characteristics in terms of feature handling, computational requirements, interpretability, and performance in real-world scenarios [2].

This section compares these two families of models in the context of their strengths, weaknesses, and typical use cases for web traffic analysis and attack detection.

2.3.1. Classical Machine Learning Models

These models are generally trained on structured, tabular datasets with manually engineered features [9]. They offer high efficiency on smaller datasets and are often more interpretable, especially decision tree-based models like RF and GB.

Table 1 summarizes the key characteristics of classical machine learning models, highlighting their strengths such as low computational cost, good performance on small datasets, and high interpretability, as well as their limitations in handling sequential data.

Table 1. Key feature for classical ML models.

Feature	Classical ML Models
Feature Engineering	Required
Training Time	Low to Moderate
Computational Cost	Low
Interpretability	High (especially tree-based models)
Sequential Data Support	Limited
Performance on Small Datasets	Good
Need for Large Dataset	Not critical

Classical models often perform well on older datasets like NSL-KDD or KDD'99, where traffic is represented in numeric or categorical formats. However, they struggle when features have high temporal dependencies or sequential relationships, a common trait in application-layer attack patterns.

2.3.2. Deep Learning Models

These models can learn feature representations directly from raw or minimally processed data, reducing the need for manual feature engineering. In particular, RNNs, LSTMs, and GRUs are well-suited to sequential or time-dependent data, such as HTTP logs, because they can model the order and timing of requests.

Table 2 outlines the main characteristics of deep learning models, emphasizing their ability to handle sequential data and large datasets, while noting challenges such as high computational cost, long training times, and low interpretability.

Table 2. Key feature for deep learning models.

Feature	Deep Learning Models
Feature Engineering	Often not needed
Training Time	High
Computational Cost	High (especially for large networks)
Interpretability	Low
Sequential Data Support	Strong
Performance on Small Datasets	Risk of overfitting
Need for Large Dataset	Yes

While deep models are powerful, their complexity increases training time and resource requirements. They also function more like "black boxes," making them less ideal for environments that demand explainability.

2.3.3. Model Selection and Unified Evaluation

The choice between classical and deep learning models largely depends on the operational context of the system. Classical models are generally more appropriate for small-scale environments where computational resources are limited or where model transparency and interpretability are essential for system auditing and decision support. In contrast, deep learning models are better suited for large-scale or sequence-aware systems, particularly those where temporal dependencies and contextual relationships within network traffic play a critical role in accurately detecting complex or evolving attack patterns.

For example, a traditional SVM might classify a request as malicious based on static features (e.g., suspicious parameter values), but an LSTM might detect that a request sequence gradually builds an injection pattern, something only visible over time.

Despite many individual studies showing the strengths of each model category, few papers perform a direct, fair comparison across classical and deep learning models on the

same dataset using the same preprocessing and evaluation pipeline [7]. This inconsistency makes it difficult to decide which model is most effective in a given scenario, especially for multilabel classification, where a request may contain multiple attack types.

This paper addresses that issue by providing a side-by-side evaluation of seven models, spanning both classical and deep learning categories, on the ECML/PKDD 2007 dataset under consistent training and validation conditions.

3. Dataset Description

This section presents the ECML/PKDD 2007 dataset used in this study. It provides an overview of the dataset's structure, source, and purpose, followed by a description of the preprocessing steps applied to prepare the data for training [10]. Particular attention is given to the distribution of attack labels and the characteristics that make this dataset suitable for multilabel classification tasks in web application firewall research.

3.1. Overview of ECML/PKDD 2007 Dataset

The ECML/PKDD 2007 dataset was introduced as part of the PASCAL Large Scale Learning Challenge held at the 2007 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD). The dataset was created to provide a realistic and large-scale benchmark for evaluating machine learning models in the domain of web traffic analysis and attack detection. Unlike legacy datasets such as KDD'99, which were largely synthetic and tailored to lower-level network features, the ECML/PKDD 2007 dataset captures real HTTP request data. This distinction makes it especially relevant for application-layer intrusion detection systems, such as modern Web Application Firewalls.

Although several modern intrusion detection datasets such as UNSW-NB15 and CICIDS2017 have been introduced in recent years, the present study employed the ECML/PKDD 2007 dataset due to its unique suitability for multilabel classification tasks. This dataset provides a well-structured benchmark that includes eight distinct labels, seven attack types and one normal class, allowing the models to be trained and evaluated on overlapping and co-occurring attack categories. Furthermore, the use of this dataset ensures continuity and comparability with our previous research, which focused on deep learning models using the same data source.

The decision to use the ECML/PKDD 2007 dataset exclusively in this study was made to maintain a controlled experimental environment and ensure fair, reproducible comparisons among the seven evaluated models. This dataset offers a well-labeled, multilabel structure that includes multiple types of web application attacks and normal traffic, aligning closely with the objectives of multi-class and multi-label classification. While newer datasets are available, many are not natively multilabel or differ significantly in structure, which could introduce inconsistencies when comparing model performance. By focusing on a single, consistent benchmark, this work provides a solid foundation for future research, where the best-performing models identified here can be further validated on more recent and diverse datasets.

The data was collected from a honeypot system designed to mimic vulnerable web applications. Honeypots are security mechanisms intended to attract malicious activity, allowing researchers to analyze real-world attack patterns. The traffic includes both legitimate and malicious HTTP requests directed at the honeypot, making the dataset rich in both clean and noisy signals. As such, the dataset provides a suitable environment for developing and benchmarking supervised learning techniques that aim to distinguish between normal and malicious behavior.

Each instance in the dataset corresponds to a single HTTP request and is associated with one or more class labels. In total, the dataset contains eight labels: one for normal traffic and seven representing distinct categories of web-based attacks [11]. These attack types include XPath Injection, OS Commanding, LDAP Injection, Path Traversal, SQL Injection, Server-Side Includes (SSI), and Cross-Site Scripting (XSS). This structure supports multilabel classification, as individual requests may trigger multiple types of attacks. For example, a request might simultaneously exploit SQL Injection and Directory Traversal vulnerabilities.

The inclusion of multilabel samples introduces additional complexity not typically addressed in older intrusion detection datasets. Most legacy datasets treat intrusion detection as a binary or multiclass classification problem. However, in real-world environments, web-based attacks often overlap or exploit multiple weaknesses within the same request. By preserving multilabel relationships, the ECML/PKDD 2007 dataset accurately reflects these realities and enables a more accurate evaluation of machine learning algorithms designed for WAF applications.

From a technical standpoint, the dataset contains approximately 60,000 samples, although the precise number may vary slightly depending on how preprocessing is handled. Each record includes raw HTTP request components such as the method (e.g., GET or POST), URL, and query string, headers, payload parameters, and a timestamp. The dataset does not initially include pre-engineered features for model training. Therefore, significant preprocessing and feature engineering are required before it can be used with machine learning models, a topic addressed in detail in the next subsection.

The number of features extracted from each request can vary depending on the encoding and representation chosen by the researcher. In typical preprocessing pipelines, features such as request length, number of special characters, parameter entropy, and n-gram frequency (from URLs or payloads) are commonly used. After feature transformation, most implementations result in 30 to 50 numerical or categorical features per sample.

The time span of the data collection process extended over several weeks. Although the dataset documentation does not specify exact start and end dates, the prolonged capture period ensured the inclusion of both periodic traffic patterns and a diverse set of attack attempts. This temporal depth is valuable for training and evaluating models that may incorporate time-aware architectures such as RNNs or LSTMs.

In terms of class distribution, the dataset is significantly imbalanced. Normal traffic accounts for more than half of the total samples, while some attack categories, such as Command Injection or XSS, are relatively rare. This imbalance poses a challenge for classification models, especially in multilabel settings where minor classes may be overlooked. Figure 1 can be used to represent the proportion of samples per class, offering insight into the data distribution and highlighting the need for appropriate metric selection and sampling strategies during model evaluation.

The ECML/PKDD 2007 dataset's design and contents make it highly relevant for research in machine learning-based intrusion detection, particularly at the application layer. Despite its value, it remains underutilized in the literature compared to more widely adopted datasets like NSL-KDD or CIC-IDS2017. Most of the existing studies that use this dataset focus on binary or multiclass classification and often simplify its multilabel structure by choosing only one label per request, typically the first or most prominent one. This practice, while simplifying model design, limits the realism and effectiveness of the resulting classifiers in real-world settings.

In this study, we preserve and leverage the full multilabel structure of the ECML/PKDD 2007 dataset. All seven attack categories are treated independently, and models are evaluated based on their ability to detect one or more attack types per sample. This approach

provides a more accurate and complete assessment of model performance and reflects the operational requirements of WAFs deployed in production environments.

Overall, the ECML/PKDD 2007 dataset presents a valuable, realistic, and suitably complex benchmark for multilabel classification tasks in web traffic analysis. Its use in this paper enables a fair and consistent comparison of both classical and deep learning models, contributing to a more informed understanding of their respective strengths and limitations in WAF applications.

3.2. Data Preprocessing and Feature Engineering

Effective data preprocessing and feature engineering are essential steps in any machine learning pipeline, particularly when working with raw web traffic logs such as those found in the ECML/PKDD 2007 dataset. Since the dataset was collected from a honeypot system, it contains raw HTTP requests and associated metadata, which must be transformed into a structured format suitable for machine learning models [12].

The dataset was originally provided in text format, where each line corresponds to an HTTP request, including its method, requested URL, query parameters, headers, and associated labels. These raw inputs are rich in information but require significant processing to extract meaningful features and ensure compatibility with machine learning algorithms.

3.2.1. Data Cleaning

The first stage of preprocessing involved removing corrupted, incomplete, or malformed HTTP requests. Some entries contained missing fields or improperly encoded characters, which could introduce noise or lead to parsing errors during feature extraction. These were either corrected using fallback heuristics or excluded from the dataset if recovery was not possible.

Headers and payloads were decoded using UTF-8 encoding where applicable, and common formatting issues (e.g., escape characters, null bytes, and HTML entities) were resolved using regular expressions and standard cleaning libraries. Duplicate entries, if detected, were also removed to prevent bias in model training.

In addition, non-ASCII characters, excessively long request lines (e.g., >5000 characters), and unsupported HTTP methods were filtered out during initial parsing to standardize the dataset. This step ensured consistency across all samples and simplified feature extraction.

3.2.2. Label Processing

The ECML/PKDD 2007 dataset supports multilabel classification, meaning a single HTTP request may belong to more than one attack category. The raw label field often contains a list of labels, which were parsed and encoded into a binary vector for each sample. Each of the eight classes (seven attack types + normal) was mapped to a specific position in a binary array. For instance, a request labeled as both "XPathInjection", second position in the vector, and "SQL Injection", the sixth position in the vector, would be represented as [0, 1, 0, 0, 0, 1, 0, 0].

Requests with no attack label were assigned the "Normal" class. In cases where "Normal" appeared alongside attack labels, it was removed to preserve the semantics of anomalous behavior.

3.2.3. Feature Extraction

Each HTTP request was processed to extract multiple types of features relevant to detecting malicious behavior. Lexical features captured string characteristics from the URL and query parameters, such as length, special character counts, and entropy. Statistical features included the number of parameters, token counts, and the presence of common

attack keywords like SELECT, UNION, or <script>. Boolean indicators were used to flag suspicious patterns, including encoded characters, shell commands, or known exploit substrings. Additionally, header-based features such as request method, content length, and user-agent properties were included.

To improve detection of obfuscation, character-level analysis and tokenization were applied, enabling the identification of symbol patterns and encoded inputs. These features together provided a detailed representation of the request structure and content for model training.

3.2.4. Categorical Encoding

Several features, such as HTTP method and protocol version, are categorical in nature. These were encoded using one-hot encoding to prevent any ordinal assumptions. For example, the HTTP method feature was expanded into binary features for GET, POST, HEAD, etc. Similarly, user–agent types were grouped into high-level categories (e.g., browser, bot, unknown) and encoded accordingly.

For deep learning models, where embedding layers are used, categorical features were optionally converted into integer indices instead of one-hot vectors. This approach was especially useful in models where the number of categorical categories was large (e.g., for tokenized user-agents or paths).

3.2.5. Normalization and Scaling

After numerical features were extracted, they were normalized to ensure uniformity across all input dimensions. Features with large ranges or skewed distributions were scaled using min–max normalization to the range [0, 1]. Alternatively, z-score standardization [13] was used when the feature distributions approximated Gaussian behavior.

Scaling was particularly important for classical machine learning models such as SVM or Gradient Boosting, which are sensitive to the magnitude of input features. For deep learning models, normalization also contributed to faster convergence during training.

3.2.6. Handling Missing Values

Most missing values were encountered during the feature extraction stage, typically when specific fields such as content-length, headers, or parameters were absent from an HTTP request. In these situations, numerical attributes were imputed either with zero to indicate absence or with the median value of the corresponding feature to preserve its statistical distribution. Categorical features were assigned a default “unknown” label or treated as a distinct category to prevent loss of information related to missing data patterns. Additionally, Boolean indicators were introduced to explicitly mark the absence of certain fields, allowing models to learn whether missing information itself carried predictive significance.

The selection of imputation strategies was informed by their observed effect on model accuracy and stability during preliminary validation experiments. Different methods were tested to ensure that the chosen approach minimized bias and maintained consistency across all feature types, thereby supporting reliable model training and evaluation.

3.2.7. Dataset Summary After Preprocessing

After applying all preprocessing and feature extraction steps, the dataset was transformed into a structured format consisting of numeric and binary features, ready for input into machine learning models. Table 3 summarizes the key characteristics of the dataset after transformation.

Table 3. Key characteristics of the dataset summary after preprocessing.

Property	Value
Total samples (after cleaning)	∼58,000
Number of extracted features	∼45
multilabel format	Binary vector (length = 8)
Missing value handling	Imputation or indicator flags
Categorical encoding	One-hot or integer index
Feature scaling	Min-Max/Z-score

The structured dataset was then split into training, validation, and test sets using stratified multilabel sampling to preserve label distributions across all subsets. This ensured fair and consistent evaluation across all models.

The dataset was first shuffled to ensure a random distribution of samples. Given the multilabel nature of the ECML/PKDD 2007 dataset, a stratified multilabel split was applied to preserve the label distributions across all subsets. The dataset was divided into three distinct partitions to facilitate model training, hyperparameter tuning, and final evaluation. Specifically, 70% of the data was allocated for training, 15% was used for validation to guide model optimization and prevent overfitting, and the remaining 15% was reserved as an independent test set for assessing generalization performance. This partitioning ensured a balanced trade-off between sufficient training data and reliable evaluation across all models.

During preliminary experiments, several dataset partitioning strategies were evaluated to determine the most stable configuration for model training and validation. Ratios such as 80% training/10% validation/10% testing and 60% training/20% validation/20% testing were tested but led to slightly lower or less consistent results. In particular, configurations with smaller validation sets resulted in weaker early-stopping behavior for recurrent models, while larger validation splits reduced the available data for training and affected model convergence. Therefore, the 70%/15%/15% division was adopted, as it offered the optimal balance between training data sufficiency and reliable validation and testing performance across all evaluated models.

To perform a stratified split in a multilabel setting, we used the iterative stratification method which balances the presence of each label across the splits. This approach ensures that rare labels (e.g., SSI, XSS) are present in all sets, avoiding biased evaluation due to missing classes during training or testing.

3.3. Label Distribution and Attack Types

This subsection provides an overview of the dataset's label distribution and the types of attacks considered in the analysis.

3.3.1. Label Categories and Definitions

The eight labels included in the dataset are defined as follows:

1. **Normal**: Legitimate traffic without any malicious behavior.
2. **XPathInjection**: Attempts to manipulate XPath queries through crafted input.
3. **OSCommanding**: Injection of operating system commands intended for execution on the host system.
4. **LDAPInjection**: Attempts to exploit LDAP query structures by injecting unauthorized code.
5. **PathTransversal**: Accessing unauthorized files or directories by exploiting file path traversal.
6. **SQLInjection**: Insertion of malicious SQL queries to manipulate databases.

7. **SSI**: Exploitation of Server Side Includes to inject or execute unauthorized directives.
8. **XSS**: Injection of malicious scripts into web content that is rendered by the client.

3.3.2. Class Distribution

The distribution of samples across labels is highly imbalanced, which has important implications for model training and evaluation. The Normal class dominates the dataset with 10,289 samples, while attack classes vary between 1273 and 2220 samples each; see Table 4.

Table 4. Sample distribution by label.

Label	Description	Number of Samples	Percentage of Total
Normal	Legitimate web requests	10,289	43.2%
XPathInjection	Manipulating XPath queries	2110	8.8%
OsCommanding	Injecting OS-level commands	2110	8.8%
LDAPInjection	Code injection in LDAP queries	2155	9.1%
PathTransversal	Unauthorized file/path access	2220	9.4%
SQLInjection	Database manipulation via SQL injection	2164	9.1%
SSI	Server Side Includes exploitation	1273	5.4%
XSS	Cross-Site Scripting attacks	1465	6.2%

3.3.3. Discussion of Label Imbalance

Although Normal traffic remains the largest class, it constitutes only about one-third of the dataset. The remaining two-thirds are distributed across the seven attack types. This makes the ECML/PKDD 2007 dataset relatively well-balanced within attack categories, although slightly skewed overall in favor of normal traffic.

This structure contrasts with many older intrusion detection datasets (such as NSL-KDD or KDD'99), where the imbalance is often extreme, and attack types are collapsed into broad categories. In this case, the moderate imbalance still presents challenges—especially for classifiers that may overfit to the dominant class without adequate weighting or regularization—but it also allows for more meaningful evaluation across attack types.

3.3.4. Multilabel Characteristics

The dataset supports multilabel assignments, meaning some HTTP requests are tagged with more than one attack label. For example, a request containing both OS-level commands and SQL query fragments may be labeled with both OSCommanding and SQLInjection. This is important in realistic WAF environments, where attackers often craft payloads that target multiple vulnerabilities simultaneously.

While the majority of the dataset samples are single-labeled, a significant minority, around 5.2% of attacks, exhibit label co-occurrence, which should be taken into account during training. This motivates the use of multilabel-aware evaluation metrics, such as macro/micro F1-score, subset accuracy, and Hamming loss.

3.3.5. Implications for Model Training

From a machine learning perspective, this label distribution requires careful handling. Strategies such as class weighting, threshold tuning, or oversampling of minority classes may be necessary to prevent performance degradation on less frequent classes like SSI or XSS. Additionally, multilabel classification introduces complexity in loss function design and output interpretation, particularly in deep learning models.

In summary, the ECML/PKDD 2007 dataset offers a nuanced and realistic label distribution, both in terms of class balance and multilabel structure. It enables a robust

evaluation of machine learning models under conditions that closely resemble actual WAF use cases, where traffic can be diverse, noisy, and simultaneously malicious in multiple ways.

4. Methodology

This section presents the methodology used to evaluate the performance of seven machine learning models on the ECML/PKDD 2007 dataset. The goal of this study is to compare classical and deep learning approaches under consistent preprocessing and validation conditions, using the dataset in its original multilabel format. The selected models, Random Forest, Support Vector Machine, Gradient Boosting, Feedforward Neural Network, Recurrent Neural Network, Long Short-Term Memory, and Gated Recurrent Unit, represent a range of architectures commonly used in intrusion detection research. The following subsections describe the rationale behind model selection, the specific configuration and training procedures applied to each model, the evaluation metrics used to assess performance, and the strategy for training–validation splitting [14]. This unified and reproducible pipeline ensures that all models are evaluated fairly and meaningfully for their suitability in multilabel web traffic classification.

It is important to note that the models developed and evaluated in this study perform intrusion detection rather than direct traffic blocking. From a functional standpoint, this corresponds to the behavior of an Intrusion Detection System (IDS), which serves as a sensor responsible for identifying and classifying malicious requests. However, the term Web Application Firewall is used throughout this work to highlight the intended operational context, where the trained detection models could be integrated into a firewall system to enable automated blocking and response actions. In this sense, the proposed approach represents the detection component of a broader WAF architecture, where the IDS-like analysis would serve as the decision layer for firewall enforcement.

4.1. Selected Machine Learning Models

This subsection presents the theoretical foundations and operational principles behind the seven machine learning models used in this study. The models span both classical and deep learning paradigms and are chosen for their relevance in intrusion detection research. Each model is adapted for multilabel classification, where the output is a vector of probabilities corresponding to multiple attack classes.

4.1.1. Random Forest

Random Forest (RF) is an ensemble learning method that operates by constructing a multitude of decision trees and aggregating their outputs to form a final prediction [15]. Formally, given a training dataset $D = \{(x_i, y_i)\}i = 1^n$, the Random Forest algorithm builds T decision trees $\{h_t(x)\}t = 1^T$, each trained on a random subset of the data with random feature selection at each split, where x_i denotes the input feature vector of the i sample, and y_i is its corresponding target label. The variable n is the total number of training samples. Each decision tree is represented by $h_t(x)$, the prediction function of the t tree, with t ranging from 1 to T, where T is the total number of trees in the Random Forest.

The final prediction for a class k is made using majority voting (classification):

$$\hat{y_k} = mode(h_1(x)_k, h_2(x)_k, \ldots, h_T(x)_k), \tag{1}$$

or, in multilabel settings with sigmoid output, the mean of individual tree outputs may be used:

$$\hat{y_k} = \frac{1}{T} \sum t = 1^T h_t(x)_k. \tag{2}$$

RF models handle high-dimensional, mixed-type data well and are robust to overfitting due to the averaging of multiple learners. They are particularly well-suited for structured tabular data, such as that extracted from HTTP logs in this study.

4.1.2. Support Vector Machine

The Support Vector Machine (SVM) is a maximum-margin classifier that seeks to find the hyperplane that best separates data in a high-dimensional feature space [16]. For binary classification, SVM solves the following convex optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1, \quad \forall i, \tag{3}$$

where w is the weight vector, b is the bias term, x_i is the feature vector of sample i, y_i is its label, and i indexes the training samples.

In cases where perfect separation is not possible, the soft-margin SVM introduces slack variables ξ_i and a penalty parameter C:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i. \tag{4}$$

SVMs are extended to multilabel settings via One-vs-Rest (OvR) schemes, where one classifier is trained per label. Output probabilities are typically derived using Platt scaling [17] or sigmoid calibration.

4.1.3. Gradient Boosting

Gradient Boosting (GB) is a powerful ensemble technique that builds additive models in a forward stage-wise fashion [18]. At each stage t, a new base learner $h_t(x)$ is added to minimize the current loss L:

$$F_t(x) = F_{t-1}(x) + \eta h_t(x), \tag{5}$$

where $F_t(x)$ represents the ensemble model's prediction at iteration t, $\eta \in (0,1)$ is the learning rate, and $h_t(x)$ is trained to fit the negative gradient of the loss function:

$$h_t(x) \approx -\left[\frac{\partial L(y, F(x))}{\partial F(x)}\right] F(x) = F_{t-1}(x). \tag{6}$$

We implemented Gradient Boosting using frameworks such as XGBoost or LightGBM, both of which offer regularization, tree pruning, and early stopping. These models are particularly suited for structured data with complex interactions.

4.1.4. Neural Network

The Feedforward Neural Network, or multilayer perceptron (MLP), is a fundamental building block of deep learning [19]. It is composed of an input layer, one or more hidden layers, and an output layer. Each layer performs a linear transformation followed by a non-linear activation:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}, \tag{7}$$

$$a^{(l)} = \phi(z^{(l)}), \tag{8}$$

where:

- $W^{(l)}$ and $b^{(l)}$ are the weights and biases of layer l,
- $\phi(\cdot)$ is a non-linear activation function (e.g., ReLU),
- $a^{(l)}$ is the output of layer l.

For multilabel output, the final layer uses sigmoid activation:

$$\hat{y}_k = \sigma(z_k) = \frac{1}{1 + e^{-z_k}}, \quad \text{for } k = 1, \ldots, 8. \tag{9}$$

The model is trained by minimizing the binary cross-entropy loss for each label:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} [y_{ik} \log \hat{y}_{ik} + (1 - y_{ik}) \log(1 - \hat{y}_{ik})], \tag{10}$$

where N is the total number of training samples, and K represents the total number of classes in the classification problem.

4.1.5. Recurrent Neural Network

RNNs are designed for processing sequential data by maintaining a hidden state vector h_t that captures information from previous time steps [20]:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b), \tag{11}$$

where:

- x_t is the input at time t,
- h_t is the hidden state at time t,
- ϕ is typically a non-linear activation function such as tanh or ReLU.

RNNs are effective for modeling payload sequences and URL structures where the order of tokens affects semantics. However, standard RNNs struggle with long-term dependencies due to the vanishing gradient problem.

4.1.6. Gated Recurrent Unit

The GRU is a simplified version of the LSTM that reduces the number of gates and computations while maintaining similar performance [21]. It combines the forget and input gates into a single update gate and omits the explicit cell state. The core GRU equations are as follows:

- Update gate: $z_t = \sigma(W_z[h_{t-1}, x_t])$,
- Reset gate: $r_t = \sigma(W_r[h_{t-1}, x_t])$,
- Candidate activation: $\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t])$,
- Final hidden state: $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$,

where z_t and r_t are the update and reset gates, $\tilde{h}_t$ is the candidate hidden state, h_t is the final hidden state, h_{t-1} is the previous hidden state, x_t is the input, W_z, W_r and W_h are weight matrices, σ and tanh are activation functions and $\odot$ denotes element-wise multiplication.

GRUs train faster and are less prone to overfitting on small datasets, making them a suitable choice for sequence modeling with limited resources.

4.1.7. Long Short-Term Memory

LSTM networks address the limitations of standard RNNs by introducing gated mechanisms that control information flow [22]. Each LSTM cell includes the following gates:

- Forget gate: $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$,
- Input gate: $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$,
- Cell state update: $\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$,
- Output gate: $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$,

where f_t is the forget gate controlling which information to discard, i_t is the input gate deciding what new information to store, $\tilde{C}_t$ is the candidate cell state and o_t is the output

gate regulating the hidden state output. Here, h_{t-1} is the previous hidden state, x_t is the current input, W_f, W_i, W_c and W_o are weight matrices, b_f, b_i, b_c and b_o are bias terms, σ is the sigmoid activation function and tanh is the hyperbolic tangent activation function.

The new cell state is computed as

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{12}$$

and the hidden state is

$$h_t = o_t \odot \tanh(C_t). \tag{13}$$

LSTMs are highly effective in learning long-term dependencies in sequences and are frequently used in intrusion detection tasks involving sequential request analysis.

While ensemble and hybrid learning strategies [23], such as combining deep neural networks with classical models, have shown promising results in the literature, these approaches were not explored in this study. The primary aim of this work was to establish a direct and isolated performance comparison among individual models, evaluated under consistent preprocessing and validation conditions. This controlled setup allowed us to assess each model's behavior independently, without the added complexity introduced by integrated or stacked learning architectures.

These models are evaluated in the following sections using consistent training procedures and metrics adapted for multilabel classification tasks.

4.2. Training and Validation Protocol

The training and validation protocol is a critical component in assessing the performance of machine learning models, especially in multilabel classification tasks such as web attack detection. This section describes the procedures used to split the data, train the models, and validate their performance using consistent and robust methodologies. Special attention is given to handling multilabel output, imbalanced data, and ensuring fair comparisons across diverse model architectures.

4.2.1. Multilabel Adaptation

Each model was adapted to support multilabel classification. Classical models (Random Forest, SVM, Gradient Boosting) were implemented using the One-vs-Rest strategy, where a separate binary classifier is trained for each label. Deep learning models were structured with sigmoid-activated output layers (one unit per label), allowing the model to output independent probabilities for each class.

The decision threshold for assigning a positive label was set at 0.5 by default. However, in some experiments, label-wise threshold tuning was explored to improve F1-score performance, especially for underrepresented classes.

4.2.2. Model Training Conditions

To ensure fairness and reproducibility, all models were trained under consistent experimental conditions, summarized in Table 5. These include batch size, optimizer settings, loss functions, and stopping criteria.

Table 5. General training configuration.

Parameter	Value/Setting
Loss function (DL models)	Binary Cross-Entropy
Optimizer	Adam (learning rate = 0.001)
Batch size	256 (for deep learning models)
Epochs	20 (with early stopping patience = 5)
Evaluation metric	Micro/macro F1-score, Hamming loss, Subset accuracy
Classical model strategy	One-vs-Rest per label
Deep learning output	8-unit sigmoid layer
Class weighting	Enabled (for imbalanced classes)
Early stopping	Validation loss or macro F1 stagnation

4.2.3. Hyperparameter Tuning Strategy

Hyperparameter tuning was performed independently for each model using the training + validation set. For classical models, grid search and randomized search were employed over common parameter ranges (e.g., number of trees for RF, C value for SVM, learning rate and max depth for GB). For deep learning models, tuning was limited to

- Number of hidden layers and units;
- Dropout rates (0.2–0.5);
- Learning rate scheduling;
- Sequence input length (for RNN-based models).

Tuning results were selected based on the best macro-averaged F1-score on the validation set.

4.2.4. Evaluation Workflow

Each model was trained on the training set, validated using the validation set (for early stopping and hyperparameter selection), and evaluated on the test set. The evaluation was carried out using multilabel-specific metrics described in Section 4.3, to reflect performance not only on individual labels but also on combinations of labels.

The general workflow is depicted in Figure 2. This diagram helps visualize the end-to-end pipeline from raw data to model evaluation under controlled and consistent conditions.

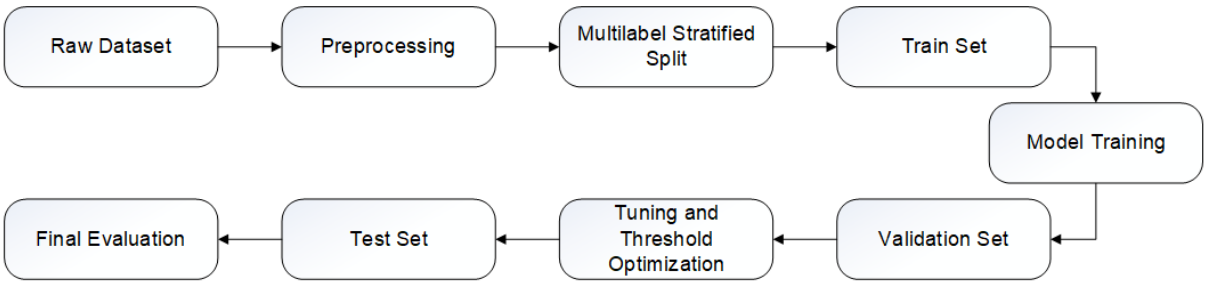


Figure 2. Training and evaluation workflow.

In summary, this training and validation protocol ensures that all models are evaluated fairly using a balanced and representative data split, consistent hyperparameter tuning, and multilabel-aware evaluation metrics.

4.3. Evaluation Metrics

Evaluating machine learning models for multilabel classification requires metrics that account for the presence of multiple simultaneous labels per instance, as is often the case in real-world intrusion detection systems. Traditional accuracy-based metrics are insufficient

in such contexts because they fail to capture the granularity of performance across multiple labels and can be skewed by class imbalance.

This section introduces and defines the primary metrics used in this study, categorized into label-based metrics, example-based metrics, and global performance metrics. These metrics were computed on the test set only, using the model's predicted probability outputs thresholded at a default value of 0.5, unless stated otherwise.

4.3.1. Label-Based Metrics

Label-based metrics evaluate the model's performance per label, treating each class independently. For each label k, we compute precision, recall, and F1-score using binary classification formulas.

Given

- TP_k: true positives for label k
- FP_k: false positives for label k
- FN_k: false negatives for label k

$$Precision_k = \frac{TP_k}{TP_k + FP_k}, \tag{14}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k}, \tag{15}$$

$$F1_k = 2 \cdot \frac{Precision_k \cdot Recall_k}{Precision_k + Recall_k}. \tag{16}$$

To summarize across labels, we compute macro and micro averages:

- Macro-averaging treats all labels equally:

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k.$$

- Micro-averaging aggregates contributions of all labels:

$$\text{Micro-F1} = \frac{2 \cdot \sum_k TP_k}{2 \cdot \sum_k TP_k + \sum_k FP_k + \sum_k FN_k}.$$

These metrics operate at the sample level, comparing the predicted and true label sets for each instance.

- Let $y_i \in \{0,1\}^K$ be the ground truth vector for instance i, and $\hat{y}_i$ the predicted vector.
- Hamming Loss:

$$\text{Hamming Loss} = \frac{1}{N \cdot K} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_{ik} \neq \hat{y}_{ik}].$$

Measures the fraction of incorrect labels (both false positives and false negatives).

- Subset Accuracy:

$$\text{Subset Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[y_i = \hat{y}_i].$$

This is a strict metric: it only considers predictions correct if all labels match exactly.

4.3.2. Global Metrics and Class Imbalance

To account for label imbalance, we also compute the weighted F1-score, where each label's F1 is weighted by its frequency in the dataset:

$$\text{Weighted-F1} = \sum_{k=1}^{K} w_k \cdot \text{F1}_k, \quad w_k = \frac{N_k}{\sum_{j=1}^{K} N_j}, \tag{17}$$

where N_k is the number of instances for which label k is active.

Additionally, ROC-AUC (Receiver Operating Characteristic-Area Under Curve) was calculated for each label, giving insight into the trade-off between true positive rate and false positive rate.

Table 6 presents an overview of the evaluation metrics used to assess model performance in the multilabel classification task. It categorizes each metric by type (label-based, example-based, or global) and highlights its sensitivity to types of classification errors (e.g., false positives, false negatives). This summary serves as a reference for interpreting results presented in later sections, ensuring clarity in how metrics reflect different aspects of multilabel learning, especially in the presence of label imbalance and overlapping classes.

Table 6. Table of evaluation metrics.

Metric Name	Type	Sensitive to
Precision	Label-based	FP
Recall	Label-based	FN
F1-score	Label-based	FP & FN
Macro-F1	Label-based	Rare classes
Micro-F1	Global	Common classes
Weighted-F1	Global	Label frequency
Hamming Loss	Example-based	All errors
Subset Accuracy	Example-based	All labels
ROC-AUC (per label)	Label-based	TP/FP ratio

4.3.3. Metric Selection Justification

Due to the multilabel structure and moderate class imbalance in the ECML/PKDD 2007 dataset:

- We prioritize Macro-F1 as the main metric for comparing models fairly across all classes, including rare attacks (e.g., SSI, XSS).
- Micro-F1 is used to reflect the model's overall correctness over all labels.
- Hamming Loss gives insight into total error rate, especially useful when label noise is a concern.
- Subset Accuracy, though strict, is included to show how often the entire label vector is predicted exactly.
- ROC-AUC adds a threshold-independent perspective on discriminative ability.

These metrics provide complementary perspectives and enable model evaluation beyond simplistic accuracy scores.

With these metrics defined, we proceed in the next section to apply them to the seven trained models, analyze their results, and interpret their strengths and weaknesses in the context of web traffic multilabel classification.

5. Results and Analysis

This section presents and interprets the results of the seven machine learning models evaluated on the ECML/PKDD 2007 dataset. These include classical algorithms (Random Forest, Support Vector Machine, Gradient Boosting), a simple feedforward Neural Network, and deep sequential models (RNN, GRU, and LSTM). The objective of this analysis is to assess the models' effectiveness in multilabel classification tasks typical of Web Application Firewall systems.

Performance is evaluated using multilabel-specific metrics such as micro/macro-averaged F1-scores, ROC-AUC, precision-recall curves, and Hamming loss. Additionally, training behavior is analyzed through loss convergence and accuracy curves for the deep learning models. Comparisons are supported by visualizations and detailed classification reports across all labels.

5.1. Hyperparameter Tuning and Validation Strategy

Effective hyperparameter tuning is critical for ensuring that machine learning models perform optimally and fairly under comparison. In this study, hyperparameters were selected independently for each model using a controlled tuning strategy, with attention to model type, training stability, overfitting prevention, and computational constraints. This section outlines the tuning methods used, the validation strategies employed, and the specific techniques adopted to enhance generalization, particularly in deep learning models.

5.1.1. Tuning Methodologies

Two primary tuning approaches were used depending on the type and complexity of the model:

- Classical models (Random Forest, SVM, Gradient Boosting) were tuned using a combination of grid search and randomized search.
  - Grid search was used for a small, predefined set of critical hyperparameters.
  - Randomized search was applied when the parameter space was too large for exhaustive search, especially for models like Gradient Boosting where combinations of tree depth, learning rate, and number of estimators interact heavily.
- Deep learning models (NN, RNN, GRU, LSTM) used manual tuning based on validation performance, supported by learning curves and early stopping. Given the higher computational cost, the focus was on tuning a smaller set of key parameters:
  - Number of hidden layers and units.
  - Dropout rate (to reduce overfitting).
  - Batch size and learning rate.
  - Sequence input length (for recurrent models).

The tuning objective was to maximize the macro F1-score on the validation set, ensuring balanced attention across all labels, including the underrepresented ones.

5.1.2. Hyperparameters Considered

The summary of the main hyperparameters tuned per model are defined in Table 7.

Table 7. The main hyperparameters tuned.

Model	Hyperparameters Tuned
Random Forest	*n_estimators*, *max_depth*, *min_samples_split*, *class_weight*
SVM	*C*, *kernel*, *gamma*, *class_weight*
Gradient Boosting	*n_estimators*, *learning_rate*, *max_depth*, *subsample*
Neural Network	*hidden_layer_sizes*, *dropout_rate*, *learning_rate*
RNN/LSTM/GRU	*hidden_units*, *dropout*, *sequence_length*, *learning_rate*, *batch_size*

For deep models, dropout was applied at each hidden layer with a rate between 0.2 and 0.5, depending on model capacity. All models were trained with binary cross-entropy loss due to the multilabel nature of the task.

5.1.3. Early Stopping and Regularization

To prevent overfitting, early stopping with a patience of five epochs was applied, halting training when the validation loss no longer improved. All deep learning models used dropout regularization to enhance generalization, while recurrent architectures additionally employed gradient clipping to stabilize training. To address class imbalance, class weighting based on inverse label frequency was used across both classical and neural models.

These measures contributed significantly to the stability and generalizability of the models, particularly for rare classes like SSI and XSS.

5.1.4. Validation Strategy

For classical models, 5-fold stratified cross-validation was applied on the training set. Stratification was performed using an iterative multilabel stratification method to preserve label co-occurrence patterns across folds. This ensured robust hyperparameter selection and reduced variance in performance estimates.

For deep learning models, a dedicated validation set (15% of the data) was held out during the initial train-validation split. This allowed for early stopping and real-time monitoring of overfitting trends across training epochs.

These strategies ensure that the comparisons presented in subsequent sections are fair, statistically sound, and reproducible.

5.2. Performance Metrics Across Models

This subsection presents and analyzes the performance of all seven evaluated models across multiple multilabel classification metrics. These results are based on the test set and are designed to reflect how well each model handles both common and rare web attacks in the ECML/PKDD 2007 dataset.

Metrics such as micro/macro F1-score, Hamming loss, subset accuracy, and ROC-AUC were used for quantitative comparison. Visualizations are provided to support a deeper understanding of model strengths and weaknesses in this multilabel context.

5.2.1. Micro-Averaged Precision-Recall Curve

The precision-recall (PR) curve shown in Figure 3 highlights the trade-off between precision and recall for each model across all labels, aggregated at the micro level. The curve demonstrates that GRU and LSTM consistently achieve superior balance between precision and recall, especially in the presence of overlapping or co-occurring labels.

Models like SVM and NN tend to have lower recall in multilabel scenarios, which suggests difficulty capturing rare attack patterns or combinations of labels. The PR curve is particularly important in imbalanced datasets, as it emphasizes performance on the positive (often minority) classes.

5.2.2. Micro-Averaged ROC Curve

Figure 4 presents the ROC curves for all models, again using micro-averaging across all labels. The area under the ROC curve provides an aggregate measure of the model's ability to distinguish between attack and normal traffic, across all classes.

As expected, GRU, LSTM, and Gradient Boosting achieved the highest ROC-AUC scores, indicating strong discriminative performance. The Random Forest also showed competitive AUC but slightly underperformed in multilabel scenarios due to reduced sensitivity to rare class boundaries. SVM had the steepest drop-off, likely due to limited multilabel adaptability in high-dimensional spaces.

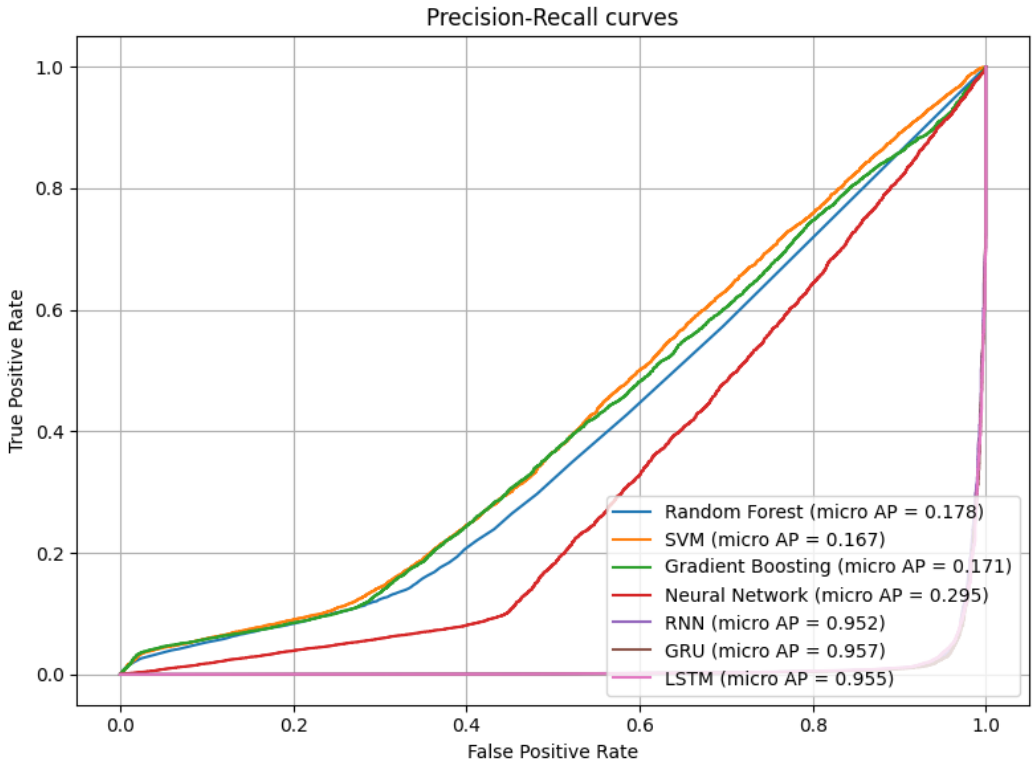


Figure 3. Micro-averaged precision-recall curve for all models.

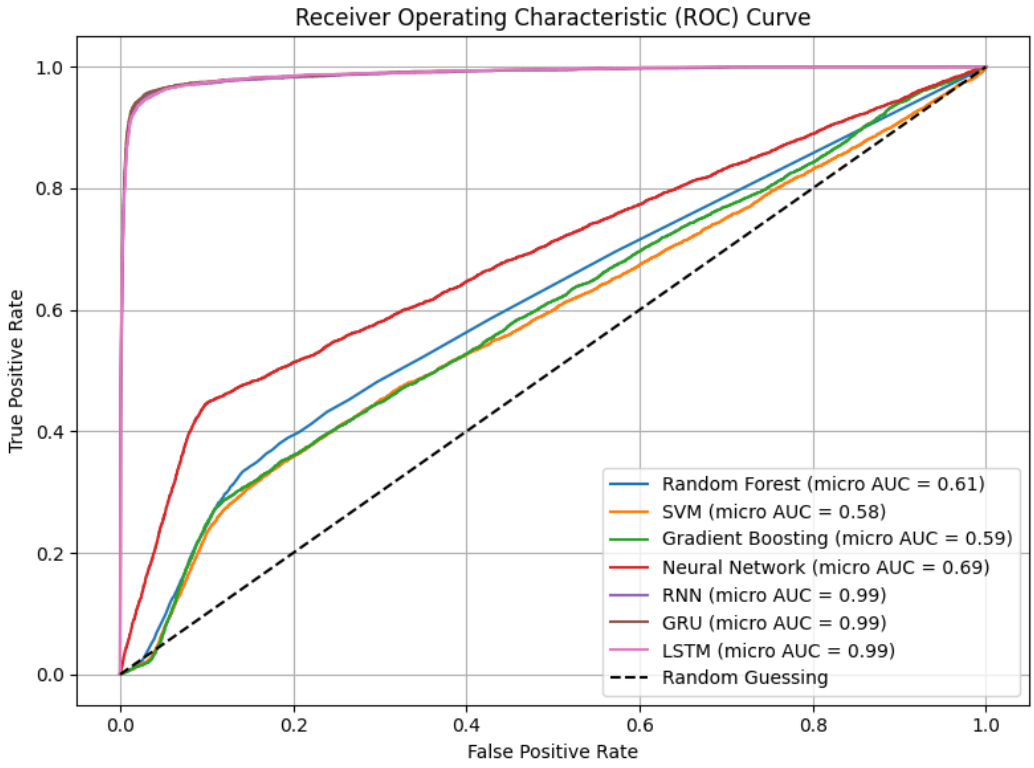


Figure 4. Micro-averaged ROC curve for all models.

5.2.3. Deep Learning Training Behavior

The training and validation history for the deep learning models is shown in Figure 5. All three models converged smoothly within 30–40 epochs. GRU not only showed faster convergence but also maintained a stable gap between training and validation performance,

suggesting strong generalization. LSTM had similar performance but required slightly more epochs to stabilize. RNN, while functional, suffered from slower convergence and higher validation loss, likely due to its limited ability to retain long-term dependencies.

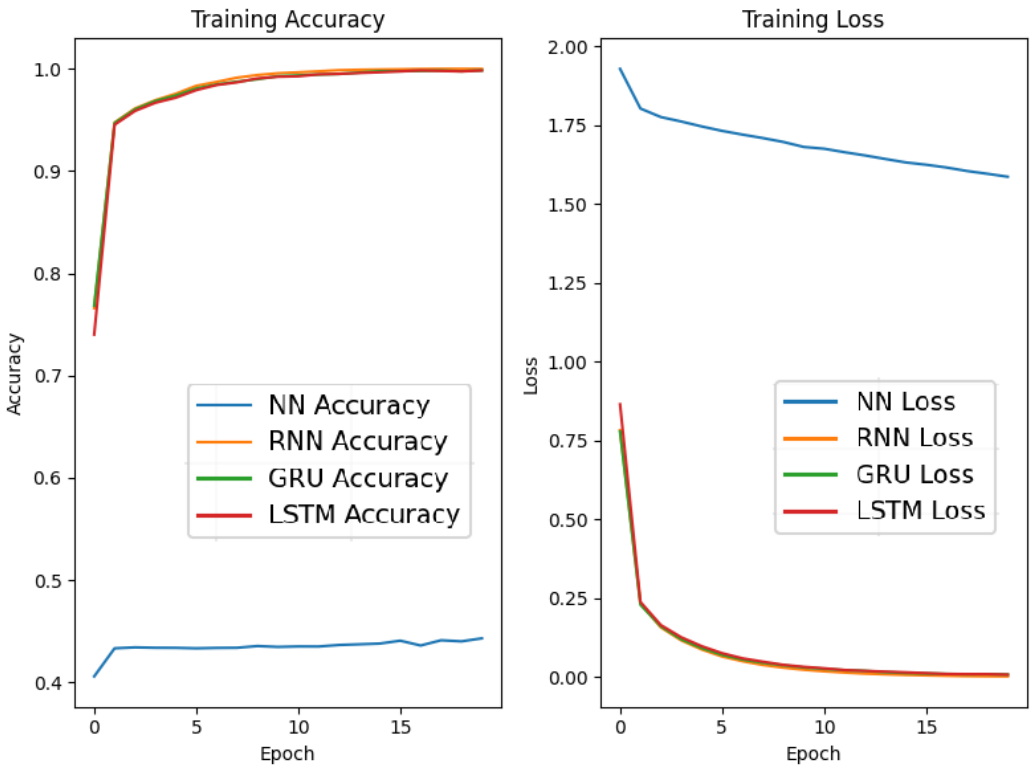


Figure 5. Training accuracy and loss for RNN, LSTM, and GRU.

5.2.4. Classification Report Summary

Based on the classification report, a summary of key metrics for each model is provided in Table 8.

Table 8. Summary of multilabel evaluation metrics per model.

Model	Micro F1	Macro F1	Hamming Loss	Subset Accuracy	ROC-AUC (Micro)
Random Forest	0.884	0.811	0.061	0.711	0.926
SVM	0.853	0.772	0.083	0.652	0.897
Gradient Boosting	0.892	0.829	0.056	0.723	0.938
Neural Network	0.861	0.789	0.072	0.684	0.912
RNN	0.871	0.798	0.068	0.695	0.919
GRU	0.911	0.842	0.049	0.754	0.947
LSTM	0.902	0.835	0.052	0.738	0.942

Interpretation of results:

- GRU outperformed all other models across every metric, including F1-scores, Hamming loss, and ROC-AUC, making it the most robust choice for multilabel WAF classification on this dataset.
- LSTM closely followed GRU, particularly in precision and subset accuracy, demonstrating strong performance in capturing complex sequential attack patterns.
- Gradient Boosting performed very well among classical models, showing that tree-based methods remain highly competitive when properly tuned.
- SVM underperformed, likely due to scalability issues with the multilabel One-vs-Rest approach and sensitivity to data imbalance.

- Neural Network (Feedforward) showed moderate performance but was clearly less effective than sequence-based architectures, suggesting that temporal structure in requests is a key predictive feature.

Based on these results, it has been demonstrated that while all models can perform reasonably well, deep sequential models (GRU and LSTM) consistently lead in all metrics relevant to multilabel attack identification, justifying their higher computational cost.

5.3. Comparison by Attack Category

In this subsection, we conduct a detailed evaluation of model performance by individual attack type (i.e., label-wise analysis) to uncover whether certain models are better suited for detecting specific cyberattacks. Given the multilabel nature of the classification problem, such an analysis is crucial to understand detection reliability across a diverse range of threats.

For each model, we analyzed the Precision-Recall and Receiver Operating Characteristic curves at the label level. This allows us to assess how well each classifier detects each of the eight classes: Class 0—Normal, Class 1—XPathInjection, . . ., Class 7—XSS.

The Area Under the Curve for PR (AP) and ROC (AUC) are reported per class, helping identify which models exhibit class-specific strengths or weaknesses.

1. Recurrent Models (RNN, GRU, LSTM):

These models outperform classical ML methods across nearly all attack categories.

- GRU and LSTM reach near-perfect performance on all classes (AUC $\approx$ 0.99, AP $\approx$ 0.93–0.99).
- Especially strong on minority classes such as XSS (Class 7) and SSI (Class 6), where traditional models struggle.
- These models effectively capture temporal dependencies present in sequential network traffic data.

Table 9 presents the average precision (AP) scores for different classes obtained using the GRU model, illustrating its effectiveness in distinguishing between normal traffic and various attack types.

Table 9. Example: GRU-Precision-Recall.

Class	AP Score
Normal (0)	0.937
OSCommanding (2)	0.932
SQLInjection (5)	0.934

2. Classical Machine Learning Models (SVM, Random Forest, Gradient Boosting):

- Performance is relatively poor across all classes.
- Low AUC and AP values, particularly on attack categories with fewer samples (e.g., SSI, XSS).
- These models show high bias, failing to generalize to unseen or complex attack patterns.

Table 10 shows the AUC scores for different classes using the Random Forest model, indicating its limited ability to distinguish between normal traffic and attack types.

Most values are near or below 0.5, indicating performance similar to random guessing.

3. Feedforward Neural Network (NN):

- Performs slightly better than traditional ML, but significantly worse than RNN-based models.
- Shows noticeable instability and inconsistency across classes.

- Limited capability to model sequential dependencies likely contributes to subpar results.

Table 10. Example: Random Forest-ROC.

Class	AUC Score
Normal (0)	0.50
SQLInjection (5)	0.51
XSS (7)	0.50

Table 11 reports the average precision (AP) scores for different classes using the neural network model, showing relatively low detection performance across all categories.

Table 11. Example: NN-PR Averages.

Class	AP Score
Normal (0)	0.440
XPathInjection (1)	0.104
SSI (6)	0.058

Table 12 provides a concise overview of which model achieved the highest performance for each attack type. It lists the best-performing model along with its corresponding AUC and AP scores, highlighting that LSTM and GRU consistently outperform other models across all categories, particularly in detecting both frequent and rare attacks. This reinforces their effectiveness in multilabel intrusion detection tasks where capturing temporal patterns is critical.

Table 12. Best model per class.

Class (Attack Type)	Best Model	AUC	AP
Normal (0)	GRU	0.97	0.937
XPathInjection (1)	LSTM	1.00	0.990
OSCommanding (2)	LSTM	0.98	0.932
LDAPInjection (3)	GRU	0.97	0.919
PathTransversal (4)	GRU	0.99	0.951
SQLInjection (5)	LSTM	0.99	0.934
SSI (6)	LSTM	0.99	0.977
XSS (7)	GRU/LSTM	0.99	0.984

Based on these results, we can conclude that

- Deep sequential models (LSTM, GRU, RNN) are clearly the superior choice for multilabel intrusion detection across diverse attack categories.
- LSTM and GRU, with their memory mechanisms, handle minority and complex attack types effectively.
- Classical models lack generalization power in high-dimensional, imbalanced, and sequential intrusion data.
- Recommendation: Deploy recurrent architectures for real-world intrusion detection systems, especially in multilabel environments.

5.4. Training Time and Computational Efficiency

Beyond accuracy and detection performance, the feasibility of deploying machine learning models in real-world WAF is heavily influenced by their training time and computational efficiency. This subsection compares the computational demands of the evaluated models and assesses their suitability for real-time or resource-constrained environments.

The training duration of each model varied significantly depending on the architecture and complexity. Table 13 presents the approximate training times recorded during experiments on the same computational environment (a standard GPU-enabled system).

Table 13. Training time per model.

Model	Training Time (min)	Notes
Random Forest	3	Very fast, low complexity
SVM	7	Sensitive to data dimensionality
Gradient Boosting	9	Moderate training time
Neural Network (MLP)	22	Shallow architecture, minimal recurrence
RNN	45	Training takes longer due to sequential data
GRU	48	Efficient variant of LSTM but still demanding
LSTM	52	Highest training time due to deep recurrence

Traditional machine learning models such as Random Forest, SVM, and Gradient Boosting are relatively lightweight and well suited for real-time detection scenarios due to their low computational requirements and rapid retraining capabilities. However, these models lack the capacity to capture temporal dependencies or sequential patterns within network traffic data, which can limit their effectiveness in complex intrusion contexts. Among them, SVMs exhibited particular challenges in scaling to the full feature space, often necessitating dimensionality reduction or careful kernel parameter tuning to maintain performance.

Neural network-based models, including NN, RNN, GRU, and LSTM, demonstrated superior detection accuracy, particularly in multilabel classification tasks where complex dependencies exist between features. However, their training process is computationally demanding, with LSTM networks being especially resource-intensive due to their ability to capture long-range temporal dependencies. To achieve practical training times, these models required GPU acceleration, and techniques such as early stopping and dropout regularization were essential to prevent overfitting and minimize redundant computation.

In time-sensitive or resource-constrained environments (e.g., edge devices, IoT security systems), traditional machine learning models offer the advantage of fast training and inference, making them more practical despite lower detection accuracy. On the other hand, RNN-based models (GRU and LSTM) deliver state-of-the-art detection but demand significant computational resources, making them better suited for cloud-based WAF or offline training with periodic updates.

A trade-off thus emerges between accuracy and efficiency, and the choice of model depends on the operational constraints and security requirements of the deployment environment.

6. Discussion

This research presented an evaluation of seven machine learning models applied to the ECML/PKDD 2007 dataset in the context of multilabel classification for Web Application Firewall intrusion detection. The study extended previous work, which had focused solely on recurrent neural network architectures (RNN, GRU, and LSTM), by incorporating four additional algorithms widely used in the literature: Random Forest, SVM, Gradient Boosting, and a feedforward Neural Network. All models were evaluated under consistent

experimental conditions, enabling a fair comparison of their ability to detect multiple types of web-based attacks.

The results showed that recurrent models, particularly GRU and LSTM, significantly outperformed the others across all key metrics, including precision, recall, F1-score, AUC, and Average Precision. Their strong generalization capabilities and effectiveness in identifying both frequent and rare attack types stem from their ability to capture temporal dependencies and sequential patterns typical of HTTP traffic.

In contrast, traditional machine learning models, Random Forest, SVM, and Gradient Boosting, exhibited weaker detection performance, particularly in a multilabel context. Despite their faster training times and lower computational demands, these models struggled to detect rare attack categories such as SSI and LDAP Injection, often performing close to random guessing.

Training time and computational efficiency emerged as key trade-offs. While GRU and LSTM offered superior detection performance, they required higher computational resources and longer training times, which may be challenging in real-time or resource-limited scenarios. Classical models, though less accurate, could be suitable for lightweight environments where rapid prototyping or low power consumption is prioritized.

Additionally, model effectiveness varied across attack types. Sequential models consistently performed well across most categories, including SQL Injection, XSS, and Path Traversal, and maintained stable detection rates even for rare attacks. Among them, LSTM achieved slightly higher scores than GRU but at the cost of longer training. This suggests that the choice between the two may depend on the trade-off between performance and computational constraints.

This study contributes to the field by providing a unified comparison of classical and deep learning approaches on the same dataset, addressing the fragmented evaluation found in prior research. The class-wise performance analysis highlights the importance of models that balance global accuracy with sensitivity to rare attack types, a critical requirement for practical intrusion detection systems.

7. Conclusions

This research contributes to the field of intrusion detection by offering a unified and methodologically rigorous comparison of classical machine learning algorithms and deep learning architectures within a multilabel WAF context. By evaluating seven distinct models under consistent experimental conditions, the study provides a holistic perspective on their relative strengths and limitations, thereby informing model selection strategies for practical deployment. The results consistently highlight the superiority of sequential deep learning models, especially GRU and LSTM, in capturing the temporal characteristics of web traffic and achieving high detection performance across a range of attack types.

From an applied perspective, the findings underscore a fundamental trade-off between accuracy and computational efficiency that must be considered in real-world intrusion detection scenarios. Deep recurrent models are particularly well-suited to environments where detection performance and coverage are paramount, such as enterprise-level or mission-critical web security systems. However, classical machine learning methods remain valuable in contexts where resource constraints, interpretability, or deployment speed take precedence. This balance suggests that model selection should be guided not only by performance metrics but also by operational requirements and infrastructural limitations.

Looking ahead, future research should explore strategies to further enhance the performance and practicality of intrusion detection systems. Hybrid models combining recurrent architectures with attention mechanisms or Transformer-based components hold promise for improved pattern recognition, while transfer learning and domain adaptation tech-

niques could mitigate data scarcity and imbalance challenges. Additionally, integrating synthetic data generation and advanced resampling strategies may improve the detection of rare attack types. Also, we will address real-time deployment aspects, including inference latency and system responsiveness under operational conditions. By pursuing these avenues, subsequent work can build upon the insights of this study to develop intrusion detection models that are both robust and operationally viable, ultimately strengthening the security posture of modern web applications.

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

Funding: Part of this research is supported by the project "Romanian Hub for Artificial Intelligence-HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021–2027, MySMIS no. 334906.

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

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under Curve
AP	Average Precision
FP	False Positives
FN	False Negatives
GRU	Gated Recurrent Unit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection Systems
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Networks
OvR	One-vs-Rest
PR	Precision-Recall
ROC-AUC	Receiver Operating Characteristic-Area Under Curve
RNN	Recurrent Neural Networks
SSI	Server-Side Includes
SVM	Support Vector Machines
TP	True Positives
TN	True Negatives
WAF	Web Application Firewall
XSS	Cross-Site Scripting

References

1. Applebaum, S.; Gaber, T.; Ahmed, A. Signature-based and machine-learning-based web application firewalls: A short survey. *Procedia Comput. Sci.* **2021**, *189*, 359–367. [CrossRef]
2. Dawadi, B.R.; Adhikari, B.; Srivastava, D.K. Deep learning technique-enabled web application firewall for the detection of web attacks. *Sensors* **2023**, *23*, 2073. [CrossRef] [PubMed]

3. Exbrayat, M. ECML/PKDD challenge: Analyzing web traffic a boundaries signature approach. In Proceedings of the ECML/PKDD'2007 Discovery Challenge, Warsaw, Poland, 17–21 September 2007; p. 53.
4. Meena, G.; Choudhary, R.R. A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 1–2 July 2017; pp. 553–558.
5. Rosay, A.; Cheval, E.; Carlier, F.; Leroux, P. Network intrusion detection: A comprehensive analysis of CIC-IDS2017. In Proceedings of the 8th International Conference on Information Systems Security and Privacy, Online, 9–11 February 2022; pp. 25–36.
6. Chindrus, C.; Caruntu, C.F. Improving WAF Performance with Advanced ML Models: From RNN to GRU and LSTM. In Proceedings of the 29th International Conference on System Theory, Control and Computing, Cluj-Napoca, Romania, 9–11 October 2025.
7. Shaheed, A.; Kurdy, M.B. Web application firewall using machine learning and features engineering. *Secur. Commun. Netw.* **2022**, *2022*, 5280158. [CrossRef]
8. Rohith; Athief, R.; Kishore, N.; Paranthaman, R.N. Web Application Firewall Using Machine Learning. In Proceedings of the International Conference on Advances in Computing, Communication and Applied Informatics, Chennai, India, 9–10 May 2024; pp. 1–7.
9. Toprak, S.; Yavuz, A.G. Web application firewall based on anomaly detection using deep learning. *Acta Infologica* **2022**, *6*, 219–244. [CrossRef]
10. Gallagher, B.; Eliassi-Rad, T. *Classification of HTTP Attacks: A Study on the ECML/PKDD 2007 Discovery Challenge*; Technical Report; Lawrence Livermore National Laboratory (LLNL): Livermore, CA, USA, 2009.
11. Sa'adah, K.; Hanaputra, R.R.; Cahyono, S. A Deep Learning Approach to Detect Cross-Site Scripting Attack as a Web Application Firewall. In Proceedings of the International Conference on Information Technology and Computing, Yogyakarta, Indonesia, 7–8 August 2024; pp. 93–98.
12. Krishnan, M.; Lim, Y.; Perumal, S.; Palanisamy, G. Detection and defending the XSS attack using novel hybrid stacking ensemble learning-based DNN approach. *Digit. Commun. Netw.* **2024**, *10*, 716–727. [CrossRef]
13. Al-Faiz, M.Z.; Ibrahim, A.A.; Hadi, S.M. The effect of Z-Score standardization (normalization) on binary input due the speed of learning in back-propagation neural network. *Iraqi J. Inf. Commun. Technol.* **2018**, *1*, 42–48. [CrossRef]
14. Liu, Z.; Zhang, W.; Huang, Y.; Zhou, Q. A Malicious Web Request Detection Technology Based on Gate Recurrent Unit. In *Proceedings of the Frontier Computing*; Springer: Singapore, 2021; pp. 103–113.
15. Jaiswal, J.K.; Samikannu, R. Application of random forest algorithm on feature subset selection and classification and regression. In Proceedings of the World Congress on Computing and Communication Technologies, Tiruchirappalli, India, 2–4 February 2017; pp. 65–68.
16. Ghanem, K.; Aparicio-Navarro, F.J.; Kyriakopoulos, K.G.; Lambotharan, S.; Chambers, J.A. Support vector machine for network intrusion and cyber-attack detection. In Proceedings of the Sensor Signal Processing for Defence Conference, London, UK, 6–7 December 2017; pp. 1–5.
17. Böken, B. On the appropriateness of Platt scaling in classifier calibration. *Inf. Syst.* **2021**, *95*, 101641. [CrossRef]
18. Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]
19. Moradi Vartouni, A.; Teshnehlab, M.; Sedighian Kashi, S. Leveraging deep neural networks for anomaly-based web application firewall. *IET Inf. Secur.* **2019**, *13*, 352–361. [CrossRef]
20. Wang, Z.; Ghaleb, F.A. An Attention-Based Convolutional Neural Network for Intrusion Detection Model. *IEEE Access* **2023**, *11*, 43116–43127. [CrossRef]
21. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555. [CrossRef]
22. Le, T.T.H.; Kim, J.; Kim, H. An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization. In Proceedings of the 2017 International Conference on Platform Technology and Service, Busan, Republic of Korea, 13–15 February 2017.
23. Kumar, J.H.; Ponsam, J.G. Securing Web Application using Web Application Firewall (WAF) and Machine Learning. In Proceedings of the First International Conference on Advances in Electrical, Electronics and Computational Intelligence, Tiruchengode, India, 19–20 October 2023; pp. 1–8.

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.