electronics

MDPI

Review

Machine-Learning-Based Traffic Classification in Software-Defined Networks

Rehab H. Serag [1,2], Mohamed S. Abdalzaher [3,*], Hussein Abd El Atty Elsayed [4], M. Sobh [1], Moez Krichen [5,6] and Mahmoud M. Salim [7,8]

1 Department of Computer and Systems Engineering, Faculty of Engineering, Ain Shams University, Cairo 11566, Egypt; 2000764@eng.asu.edu.eg (R.H.S.)
2 Department of Electronics and Communications Engineering, Faculty of Engineering, Egyptian Russian University, Badr City 11829, Egypt
3 Department of Seismology, National Research Institute of Astronomy and Geophysics, Helwan 11421, Egypt
4 Department of Electronics and Communications Engineering, Faculty of Engineering, Ain Shams University, Cairo 11566, Egypt; helsayed@eng.asu.edu.eg
5 Department of Information Technology, Faculty of Computer Science and Information Technology, Al-Baha University, Al-Baha 65528, Saudi Arabia
6 ReDCAD Laboratory, University of Sfax, Sfax 3038, Tunisia
7 Interdisciplinary Research Center for Communication Systems and Sensing, King Fahd University of Petroleum & Minerals (KFUPM), Dhahran 31261, Saudi Arabia
8 Department of Electronics and Communications, Faculty of Engineering, October 6 University (O6U), Giza 12585, Egypt
* Correspondence: msabdalzaher@nriag.sci.eg

Abstract: Many research efforts have gone into upgrading antiquated communication network infrastructures with better ones to support contemporary services and applications. Smart networks can adapt to new technologies and traffic trends on their own. Software-defined networking (SDN) separates the control plane from the data plane and runs programs in one place, changing network management. New technologies like SDN and machine learning (ML) could improve network performance and QoS. This paper presents a comprehensive research study on integrating SDN with ML to improve network performance and quality-of-service (QoS). The study primarily investigates ML classification methods, highlighting their significance in the context of traffic classification (TC). Additionally, traditional methods are discussed to clarify the ML outperformance observed throughout our investigation, underscoring the superiority of ML algorithms in SDN TC. The study describes how labeled traffic data can be used to train ML models for appropriately classifying SDN TC flows. It examines the pros and downsides of dynamic and adaptive TC using ML algorithms. The research also examines how ML may improve SDN security. It explores using ML for anomaly detection, intrusion detection, and attack mitigation in SDN networks, stressing the proactive threat-detection and response benefits. Finally, we discuss the SDN-ML QoS integration problems and research gaps. Furthermore, scalability and performance issues in large-scale SDN implementations are identified as potential issues and areas for additional research.

Keywords: software-defined networking (SDN); machine learning (ML); Quality of Service (QoS); traffic classification (TC); security

check for updates

Citation: Serag, R.H.; Abdalzaher, M.S.; Elsayed, H.A.E.A.; Sobh, M.; Krichen, M.; Salim, M.M. Machine-Learning-Based Traffic Classification in Software-Defined Networks. *Electronics* **2024**, *13*, 1108. https://doi.org/10.3390/electronics13061108

Academic Editor: Christos J. Bouras

Received: 5 February 2024
Revised: 11 March 2024
Accepted: 13 March 2024
Published: 18 March 2024

Copyright: © 2024 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

The installation and configuration of network elements are complex tasks that require skilled personnel. When dealing with network nodes that interact with each other in complicated ways, a system-based approach involving simulation is necessary. However, the current programming interfaces of most networking equipment make it difficult to achieve this [1]. Furthermore, to manage large, multi-vendor networks, with various technologies becoming increasingly costly, service providers face resource shortages and

rising real-estate expenses. A novel network paradigm is required to integrate network management and provisioning across many domains [2].

In network devices like switches and routers, SDN is a technique that separates the control plane from the data plane [3,4]. The control plane and data plane are tightly entwined in conventional networks, making it challenging to manage and scale the network [5]. In an SDN design, a central controller controls the network and communicates with switches and routers using a standard protocol, such as OpenFlow protocol [6].

Increased network scalability and flexibility are advantages of SDN. Network administrators may simply manage, configure, and enhance the network with a centralized controller. SDN additionally enables the development of virtual networks that can be altered to accommodate particular applications or traffic types. The SDN architecture is depicted in Figure 1, and is made up of the data plane, control plane, and application plane [7,8].

The data plane consists of network devices such as routers, switches, and access points that are accessed and managed through control–data-plane interfaces (C-DPIs) by SDN controllers. The most commonly used C-DPI is the OpenFlow protocol [6,9]. The implementation of the SDN architecture heavily relies on the control plane. Essentially, the control plane functions as a separate process that operates within the control layer. This layer consists of one or more controllers that offer a comprehensive perspective of the entire SDN system through C-DPI. The controllers consist of essential components, such as a coordinator and virtualizer, which are responsible for managing the behavior of the controller. Additionally, there is a control logic that translates the networking needs of applications into instructions for allocating network element resources. Finally, the application plane is made up of one or more network applications that communicate with the controller(s) in order to use an abstract view of the network for internal decision-making. These applications exchange data with the controller(s) using an open application–controller-plane interface (A-CPI), such as REST API [9].

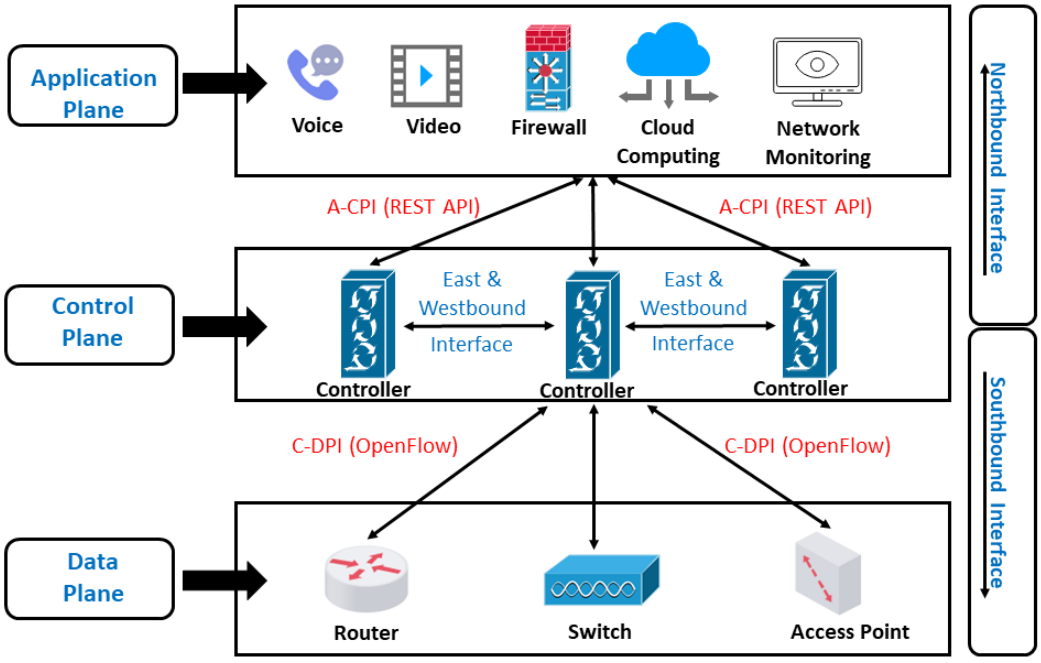


Figure 1. SDN architecture.

Table 1 presents the common existing SDN controllers: NOX [10], Floodlight [11], POX [12], OpenDayLight [13], RYU [14], and Beacon [15]. These controllers can be categorized as either centralized, in which a single control entity manages the entire network, or distributed, in which the network is divided into various sections for management [16,17].

Centralized controllers can be classified as either physically centralized or logically centralized. Physically centralized controllers are installed on a single server and are

responsible for managing the entire network. The benefit of a physically centralized controller is its ease of use and management due to having only one controller [11]. A logically centralized controller utilizes numerous physical servers, with each controller of a specific network duty. They all, however, use a centralized data store to replicate a common network state [18].

Distributed controllers serve as a distributed control plane for network management. Nevertheless, the network is partitioned into multiple domains, with each domain being managed by its own controller [19,20]. Distributed controllers come in two forms: flat and hierarchical designs. In a flat design, the network is divided into separate domains, with each domain having its controller. Controllers utilizing the flat design communicate with each other using east–west interfaces to gain a global network view. In contrast, hierarchical design employs a two-layer controller model. The first layer is a domain controller that handles switches and runs applications in its local domain, while the second layer is a root controller that maintains the global network and manages the domain controllers [21].

Table 1. A summary of controller types and programming platforms used.

Controller	Programming Language Used	Created by	Architecture
NOX	Python	Nicira	Physically centralized
FloodLight	Java	Big Switch Networks	Physically centralized
POX	Python	Nicira	Physically and logically centralized
OpenDaylight	Java	Linux Foundation	Distributed flat design
Ryu	Python	NTT Labs	Physically centralized
Beacon	Java	Standford University	Physically centralized

One of the most important aspects of SDN is that it allows for network programmability, which enables the seamless integration of artificial intelligence (AI) into communication networks. By leveraging the application programming interface (API), SDN empowers network managers to send powerful programming instructions to network devices. With the help of AI, it can not only schedule automated and intelligent business orchestrators but also develop AI-optimized network strategies and automatically convert them into task scripts, which can be assigned to network allocation tasks via the API. Additionally, network statistics information can be automatically collected and processed to provide a solid foundation for ongoing network optimization. New functionalities can also be intelligently added as needed to the network environment via SDN applications [22].

Machine learning (ML) is a crucial tool for enabling AI [23] as it can effectively predict and schedule network resources based on the available data inputs [24,25]. It has applications in various areas providing data acquisition and analysis by emulating human learning behavior of knowledge [26]. ML aims to enable computers to determine and enhance their performance over time without being explicitly programmed to do so [27]. ML algorithms can be supervised, unsupervised, semi-supervised, or reinforcement, varying based on the type of data utilized for model training [28,29]. Supervised learning (SL) is the process of training a model using labeled data when the right output for each input is known. Unsupervised learning (USL) includes finding patterns and relationships in unlabeled data. Semi-supervised learning (SSL) is a set of both SL and USL. In reinforcement learning (RL), an agent learns to act in a given environment in order to maximize a reward [30].

Network managers may therefore create networks that are more flexible, efficient, and safe by integrating SDN and ML. According to Figure 2, In SDN, a variety of tasks can benefit from the utilization of ML algorithms, such as network resource management, where they can forecast traffic demand and dynamically assign network resources to

satisfy it. This may result in greater network resource use, which would lower overall operation costs [26].

By examining user behavior, network anomalies, and traffic patterns, ML can be used to find potential security vulnerabilities [31]. This can lessen the threat of cyberattacks, particularly from malware, which is known for its ability to remain undetected in systems and execute automated coordinated attacks, making it particularly destructive for distributed systems such as IoT and Smart cities [32]. By providing real-time detection and mitigation assistance, this approach enhances cybersecurity measures. Additionally, ML can support the detection and isolation of network defects as well as the prediction of network performance decline, resulting in a more effective and dependable network [22,33].

Last but not least, ML can be used to categorize network traffic according to the kind of application or user behavior, enabling the prioritizing of high-priority traffic and assisting in making sure that vital applications obtain the necessary QoS levels. This can improve customer pleasure and experience, especially in applications that need real-time replies, high throughput, or low latency [34].

In conclusion, ML has the potential to be a potent tool for improving a number of SDN-related features, such as security, resource management, routing optimization, QoS prediction, and TC. Organizations may optimize their networks for greater performance, dependability, and security by utilizing ML techniques, which will ultimately improve their business outcomes.

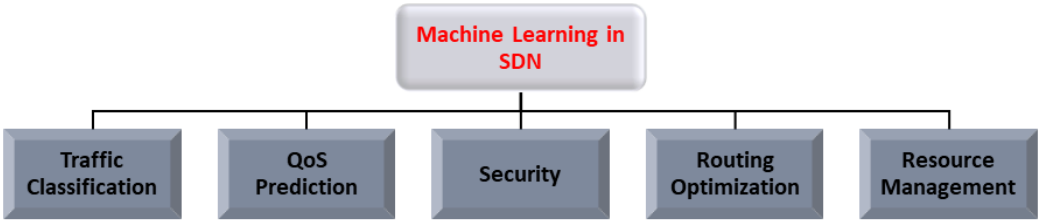


Figure 2. Machine learning applications in SDN.

Additionally, the SDN architecture's centralization and programmability, as well as the controller's capacity to gather real-time data, allow for the application of "intelligence" via ML approaches for effective routing and QoS provisioning [35].

SDN and ML have the ability to work together to build extremely intelligent and effective networks that can accommodate changing situations while delivering greater performance and security. We may anticipate seeing many more potential uses of this technology in the networking industry as ML develops.

1.1. Motivation

In [1], the focus is on the initial efforts to examine how AI is applied in the context of SDN. However, it is noteworthy that this paper does not specifically delve into TC in SDN using ML methods, but rather explores broader applications and implications of AI within the SDN framework. The overview presented in [33] provides a highly detailed introduction to basic ML algorithms and their applications in SDN networks, offering valuable references and guidance for further study. However, it is important to note that this paper covers studies only until 2018; thus, newer developments and advancements in the field may not be fully captured. The survey conducted by [26] serves as an introduction to relevant studies exploring the intersection of ML algorithms and SDN network applications, providing insights into their combined impact and potential in the field. While it may provide insights into the combined impact and potential of ML algorithms in SDN, it likely does not delve deeply into TC using ML methods. In [36], the focus is on IP TC using ML, although it does not delve into TC within the context of SDN. Our primary research objective is to offer a comprehensive overview of TC using ML techniques specifically applied in the context of SDN.

1.2. Contribution

The contributions of the paper can be listed as follows:

- Exploration of ML techniques for TC in SDN environments in a comprehensive manner.
- Incorporating the most recent research efforts in the SDN TC field.
- Including the most recent publicly available datasets suitable for training and evaluating ML models in SDN TC tasks.
- Highlighting the role of the ML model for mitigating the SDN security aspects.
- Discussing the limitations and open research issues in SDN TC.
- Providing insights into areas requiring further investigation and development.

Our paper is organized as follows. First, QoS in SDN using ML is discussed in Section 2. In Section 3, a comparison between traditional and ML TC methods is provided. Section 4 presents SDN TC using ML. Security in SDN using ML is presented in Section 5. Section 6 contains some useful datasets. Limitations and open research issues are introduced in Section 7. Finally, the paper is concluded in Section 8.

2. QoS in SDN Using Machine Learning

QoS is the ability of a network to give priority to selected network traffic and provide better service to users by ensuring dedicated bandwidth, controlling jitter and latency, and enhancing loss characteristics. QoS aims to provide end-to-end guarantees, and there are multiple technologies available to achieve this, which can be used individually or in combination. Resource reservation and allocation, prioritized scheduling, queue management, routing, and other services can be utilized by a network operating system to implement QoS.

Initially, the traditional network was not designed with QoS in mind, and various techniques were later introduced to improve performance tuning. These techniques allowed Internet Service Providers (ISPs) to optimize the internet as required. However, with emerging technologies like big data, cloud computing, and an increasing number of devices, the traditional internet faces new challenges that it struggles to cope with. SDN addresses these issues by making the internet more flexible and programmable [37]. So, as mentioned above, QoS refers to the ability to prioritize network traffic based on its importance and ensure that critical traffic receives preferential treatment over non-critical traffic. TC is one method that can be used to achieve this prioritizing [38,39].

In SDN, TC is often carried out by the controller, which can make use of ML algorithms to automatically recognize and categorize distinct forms of network traffic based on characteristics like packet size, protocol type, and application behavior. The controller can then apply QoS policies, such as giving priority to important traffic or limiting the bandwidth of specific categories of traffic, using this information.

3. Traffic Classification Using Traditional Methods vs. Machine Learning

3.1. Traditional Methods

In computer networks, traditional methods for TC imply the utilization of specific signatures, ports, or protocol headers to distinguish the traffic type. These methods are based on predefined rules that are used to distinguish between different types of traffic. Some of the commonly used techniques for TC include port-based, payload-based, deep packet inspection (DPI), and statistical-based techniques [40].

3.1.1. Port-Based TC

In the past, a widely adopted approach was port-based classification, which achieved some degree of success due to the prevalence of fixed port numbers assigned by the Internet Assigned Numbers Authority (IANA) [41]. However, this strategy revealed significant drawbacks over time. For instance, numerous applications emerged that did not possess registered port numbers, and many of them utilized dynamic port negotiation techniques

to evade firewalls and network security measures. Additionally, the utilization of IP-layer encryption, obfuscation, and proxies can obscure the TCP or UDP header, rendering the original port numbers undetectable [42,43].

According to [44], utilizing the IANA list, port-based techniques achieved no more than 70% accuracy. Similarly, [45] discovered that such techniques were unable to identify 30–70% of the traffic flows they examined.

3.1.2. Deep Packet Inspection or Payload-Based TC

The deep packet inspection (DPI) technique, also known as the payload approach, was proposed to overcome the limitations of port-based classification techniques [43]. DPI classifies traffic by analyzing packet payloads and matching them with known protocol signatures [46–49]. Protocol signatures are established using regular expressions and evaluated by automata sequentially, requiring significant memory resources. Additionally, DPI is executed within the communication path, which can lead to scalability issues [42].

DPI tools such as L7-filter and OpenDPI [50,51] have been widely employed. In order to evaluate DPI techniques, in [52], it was discovered that even popular tools such as the L7-filter were only able to correctly classify 67.73% and 58.79% of bytes on the UNIBS and POLITO data sets, respectively.

Maintaining up-to-date signatures is essential for DPI techniques to remain effective, but this often requires manual effort. Unfortunately, as network applications continue to evolve, obtaining accurate signatures can become increasingly difficult [43]. In addition, introducing devices that support DPI into a network can be a costly and complex process. Also, DPI is often difficult or impossible to perform when working with encrypted traffic [33]. Furthermore, as network applications continue to proliferate rapidly and many of these applications offer similar services in practice, their QoS requirements tend to be alike. Attempting to identify each specific application using DPI becomes inefficient. Additionally, maintaining a database containing all web applications is impractical. In an operational SDN network, TC must be real-time and cost-effective. Utilizing simple DPI technology can exhaust significant controller computing resources and introduce noticeable delays to the network, thereby reducing network responsiveness [53,54].

3.1.3. Statistical-Based TC

This technique can categorize traffic streams by analyzing their statistical properties at the network layer rather than thoroughly examining the packet contents. It operates on the assumption that traffic with similar QoS requirements has comparable statistical features. As a result, several source applications can be recognized. The approach can classify flows into clusters with similar patterns by detecting trends in their properties such as the size of the initial few packets, arrival timings, packet length, IP address, round trip time, and source/destination ports [55,56].

3.2. TC Using Machine Learning

To overcome the limitations of traditional TC methods, ML algorithms are used [57,58]. The study and development of algorithms that can learn complicated correlations or patterns from empirical data, allowing them to make reliable decisions, is essential to the discipline of ML [59].

Figure 3 shows that ML typically involves several phases, including preprocessing, training, and testing. During preprocessing, data are prepared and processed, which can involve tasks such as filtering, imputation, and tuning for specific purposes. After preprocessing, the data are used to train ML methods. Finally, the system uses the trained data to make decisions based on input received during the training phase [39,56].

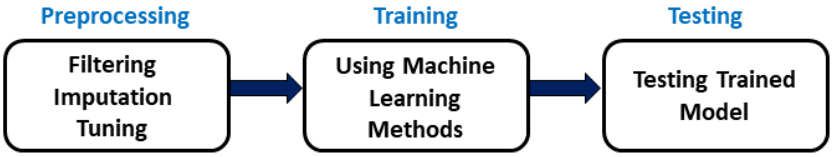


Figure 3. Machine learning phases.

ML algorithms exhibit variations in their methodology, and we classify them into four distinct categories according to the nature of the data they handle, the output they generate, and the specific task or problem they aim to address: supervised learning (SL), semi-supervised learning (SSL), unsupervised learning (USL), and reinforcement learning (RL) [26,42].

3.2.1. Supervised Learning

SL algorithms construct a mathematical model using a labeled training dataset that includes both inputs and their known outputs. These data are used by the algorithms to build a model that depicts the learned relationship between the input and output. Once trained, the model can be used to predict the output for new input data [60,61]. SL has become increasingly popular and is used in a diverse array of applications, including spam detection, speech, and object recognition [62]. SL can include both classification algorithms, which are used to predict discrete variables, and regression algorithms, which are used to predict continuous variables. These algorithms are acquired from the data and have the capability to generate predictions for novel, unseen data [26]. The drawback of the SL method is that it can attain a high level of accuracy in classifying known applications but is unable to identify unknown ones. Nonetheless, obtaining accurately labeled data can sometimes be challenging [43]. Additionally, SL not only requires a large number of data, but they must also be labeled [63]. An overview of some commonly used SL algorithms such as Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), and Key Nearest Neighbor (KNN) will be provided.

- Decision Tree (DT)
  This method is represented by a tree structure in which the database features are represented by internal nodes and branches that specify decision criteria, and the outcome is expressed by each leaf node in a tree structure [64]. This approach computes entropy on the dataset to find information gains and classify the data. It computes entropy on the dataset to determine the root node with the maximum information gain. The technique is then repeated to separate branches and finish the tree [63].
  DT has several advantages, including its simplicity of interpretation and visualization, its ability to implicitly perform feature selection, and its ability to handle non-linear relationships among parameters. However, DT can be prone to overfitting the training data and can generate overly complex trees. It is also susceptible to instability as even little variations in the data lead to the generation of a fully contrasting tree. Furthermore, it is prone to instability because even minor changes in the data might result in the construction of an entirely different tree. Furthermore, DT may struggle to manage complex systems with inconsistent features [65,66].
- Random Forest (RF)
  The RF method is used to prevent over-fitting in the DT algorithm [33]. An RF is made up of numerous decision trees that are combined to produce a stable and trustworthy forecast. This prediction is then used for training and class prediction [67–69]. Each DT in the RF makes a class prediction, and the model prediction is made by choosing the class that has received the greatest number of votes [70].
  In addition to their capacity to manage noisy and correlated datasets and their capacity to increase classification accuracy, RF algorithms have a number of benefits [71]. In comparison to DT algorithms, they are also less prone to overfitting. In addition to offering very effective classification models, RF has the ability to assess the significance

and effects of each variable utilized in the classification procedure [72]. Additionally, it is capable of handling big datasets and missing values [63].
On the other hand, RF suffers from some disadvantages. Increasing the number of trees in RF can enhance prediction accuracy; however, this may lead to longer training times and higher memory requirements due to the large number of trees utilized. Also, RF may not produce accurate results for datasets with small sample sizes or low-dimensional data [73].

- Support Vector Machine (SVM)
  SVM is a widely used SL technique that was created by Vapnik and others [74]. It is a type of common linear classifier that implements binary classification.
  SVM seeks to find a feature space separation hyperplane that maximizes the margin between separate classes. It is worth noting that the margin refers to the distance between the hyperplane and the nearest data points of each class, and these data points are known as support vectors [75–77].
  SVM is a reliable algorithm that produces fewer false alarms in binary classification jobs. Its detection system can significantly reduce the amount of time necessary for attack identification and classification observation. Furthermore, when SVM is applied at the SDN controller level, its complexity shows a small impact on the total SDN framework [26].
- K Nearest Neighbor (KNN)
  KNN is an SL approach that employs the k nearest neighbors of an unclassified sample to determine its classification. As shown in Figure 4, the KNN algorithm is as follows: "if the majority of the k nearest neighbors belong to a particular class, the unclassified sample is classified into that class" [78]. So the detailed steps are

  1. Determine the value of the parameter "K" representing the number of neighbors.
  2. Calculate the Euclidean distance for the K neighbors.
  3. Identify the K nearest neighbors by considering the computed distance.
  4. Tally the occurrences of data points for each category within the K neighbors.
  5. Allocate the new data point to the category that has the highest frequency among the K neighbors.

  KNN is easy to implement, has high accuracy, calculates the features easily, and is suitable for multiclass classifications. However, for large datasets, KNN can be time-consuming [26].

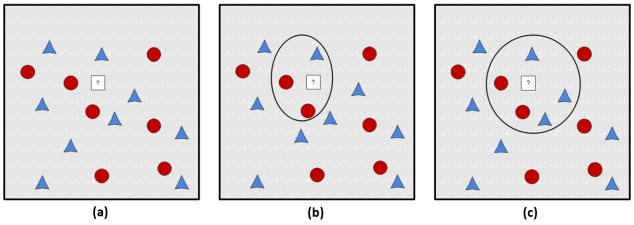


Figure 4. KNN algorithm example: (**a**) Before KNN ("?" represents the unclassified sample). (**b**) For K = 3, for the three nearest neighbors, one of them is classified as belonging to the class "Blue" while the remaining two neighbors are classified as belonging to the class "Red". As a result, the unclassified example will be categorized as class the "Red". (**c**) For K = 5, For the three nearest neighbors, three of them are classified as belonging to the class "Blue" while the remaining two neighbors are classified as belonging to the class "Red". As a result, the unclassified example will be categorized as class "Blue".

- Naïve Bayes (NB)
  NB is referred to as a probabilistic classifier because it depends on Bayes' Theorem. Bayes' theorem applies conditional probability to compute the chance of an event taking place based on the prior knowledge of conditions that may correlate with the event. Naïve Bayes theory can be represented as

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \quad (1)$$

where X and Y are events, P(X) is the prior probability of event X independent of event Y, P(Y) is the probability of event Y, P(X | Y) is called the posterior probability and it is the probability that event X will occur given the condition that Y is true, P(Y | X) is called the likelihood of X given fixed Y and can also be known as the probability that event Y will occur given the condition that X is true [79]. Those probabilities are calculated based on the training set. When classifying a fresh input data sample, the probability paradigm may generate different posterior probabilities for each class. The sample will be divided into classes based on the class with the greatest likelihood of succeeding.

The good side of Baye's theorem is that it only requires a dataset of a small size to learn the probability paradigm. On the other hand, it assumes that its predictors are conditionally independent, meaning they are not associated with any of the other features in the model. Additionally, it assumes that all features have an equal impact on the outcome. However, these assumptions are frequently not met in real-world situations [80].

3.2.2. Unsupervised Learning

To overcome the drawbacks of SL, USL is used. USL is utilized for clustering and data-aggregation tasks [61,81], where the data provided to the learner are unlabeled. In such scenarios, algorithms group the data into distinct clusters based on similarities found in the feature values [82].

USL models are required to establish relationships among elements in a dataset and classify raw data without external assistance [83]. USL algorithms can automatically discover patterns within unlabeled datasets. Nevertheless, the constructed clusters must still be mapped to their corresponding applications. Since the number of clusters is typically much greater than the number of applications, this can pose a challenge for TC tasks [43]. The algorithms commonly utilized in USL include K-Means and Self-Organizing Map (SOM).

- K-Means
  K-means is an unsupervised ML algorithm used for clustering data into groups or clusters based on similarity. The objective is to divide the data into K clusters, where each data point is assigned to the cluster whose mean or centroid is closest. The algorithm iteratively assigns data points to the nearest cluster centroid and adjusts the centroids until convergence, reducing the variance within each cluster. K-means finds applications in diverse fields like customer segmentation, image segmentation, and anomaly detection [84].
- Self-Organizing Map (SOM)
  A self-organizing map (SOM) serves as a USL technique utilized for reducing dimensionality and visualizing data. It functions by projecting high-dimensional input data onto a lower-dimensional grid of neurons, where each neuron represents a prototype or cluster within the original input space [85,86].
  In the SOM algorithm, neurons within the grid are organized based on similarities present in the input data. During the training process, input vectors are introduced to the network, and the neuron that closely matches the input vector is identified using a similarity metric, often derived from Euclidean distance. Subsequently, the weights of the winning neuron and its neighboring neurons are adjusted to move closer to the input vector, facilitating the self-organization of the map [87].
  SOMs offer a valuable means of visualizing high-dimensional data in a lower-dimensional space while retaining the topological characteristics of the original input space. They find applications across various domains, including data visualization, clustering, and pattern recognition [88].

3.2.3. Semi-Supervised Learning

Traditional ML technology is classified into two types: SL and USL. SL employs labeled sample sets for learning, while USL uses only unlabeled sample sets. However, in practical situations, the cost of labeling data can be very high, resulting in limited availability of labeled data, while a considerable number of unlabeled data are easily accessible [26]. Consequently, SSL techniques have gained popularity and are evolving rapidly, as they can utilize both labeled and unlabeled samples [89].

Typically, an SSL algorithm consists of two stages: the initial step involves analyzing labeled data to generate a general rule, which is subsequently utilized to deduce unlabeled data. Currently, the performance of SSL techniques is inconsistent and requires further enhancement [83]. Pseudo labeling [90,91], Expectation Maximization (EM), co-training, transductive SVM, and graph-based approaches are examples of SSL methods [92,93].

- Expectation Maximization (EM)
  Expectation maximization (EM) is a powerful algorithm used in statistical modeling, particularly in situations where data have missing or incomplete values or when there is a need to estimate the parameters of a probabilistic model. It is an iterative method that aims to find the maximum likelihood (ML) or maximum a posteriori (MAP) estimates of parameters in probabilistic models with latent variables [94].
  The EM algorithm consists of two main steps:
  (1) Expectation (E) step: In this step, the algorithm computes the expected value of the latent variables, given the observed data and the current estimates of the model parameters. It calculates the posterior probability distribution over the latent variables using the current parameter estimates and the observed data.
  (2) Maximization (M) step: In this step, the algorithm updates the model parameters to maximize the likelihood or posterior probability of the observed data, given the expected values of the latent variables computed in the E step. It finds the parameter values that increase the likelihood of the observed data, incorporating the information from the latent variables. The E and M steps are iteratively repeated until convergence, where the algorithm reaches a point where there is no significant improvement in the model parameters or likelihood of the data [95].
- Transductive SVM
  In traditional SL with SVMs, the algorithm learns from labeled data to classify new, unseen data points. However, in transductive SVMs, the algorithm aims to label the entire dataset, including both labeled and unlabeled instances, based on the structure of the data and the provided labels [96].
  Transductive SVMs use a combination of labeled and unlabeled data to create a decision boundary that separates different classes in the data space. By incorporating information from both labeled and unlabeled instances, transductive SVMs can potentially improve the accuracy of classification, especially when labeled data are limited or expensive to obtain [97].
  The transductive learning process in SVMs involves optimizing an objective function that considers both the labeled data's class labels and the model's predictions on the unlabeled data. This optimization aims to find the decision boundary that best fits the labeled data while also considering the distribution and structure of the unlabeled data [98].
- Co-training
  Co-training is an SSL algorithm designed for scenarios where a limited number of labeled data are available alongside a large number of unlabeled data. The key idea behind co-training is to leverage the unlabeled data to improve the performance of a classifier trained on the labeled data [99].
  The algorithm typically involves two classifiers, each trained on a different subset of features or views of the data. In each iteration of the algorithm, the classifiers are trained using the available labeled data, and then they make predictions on the

unlabeled data. Instances with high-confidence predictions (i.e., predictions with high certainty) are then added to the labeled set, and the classifiers are retrained using the expanded labeled set. The co-training algorithm iterates between these steps, gradually incorporating more unlabeled data into the training process and refining the classifiers. The process continues until a stopping criterion is met, such as reaching a maximum number of iterations or when the performance of the classifiers stabilizes [99,100].

3.2.4. Reinforcement Learning

RL represents a form of ML training that relies on rewarding favorable behaviors and/or penalizing unfavorable ones. In general, an RL agent possesses the ability to perceive and interpret its environment, take actions, and acquire knowledge through the process of trial and error [101,102]. In the context of SDN implementation, RL is employed, with the controller assuming the role of the agent, while the network serves as the environment. The controller observes the state of the network and learns to make decisions regarding data forwarding. Figure 5 provides an overview of the ML algorithms.

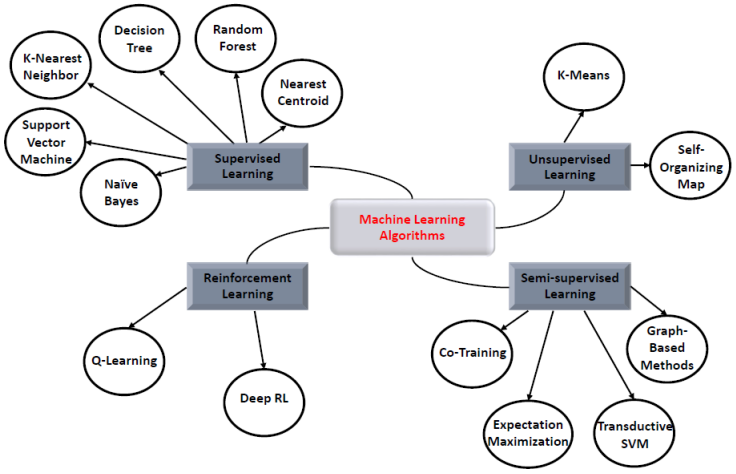


Figure 5. Machine Learning Algorithms.

3.2.5. Ensemble Learning

Ensemble learning is an ML technique that combines the predictions of multiple individual models to improve overall performance. Instead of relying on a single model, ensemble methods leverage the diversity and complementary strengths of multiple models to make more accurate predictions or decisions [103].

The primary categories of ensemble learning techniques include bagging, stacking, and boosting [104].

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique used to improve the accuracy and robustness of ML models. It involves training multiple instances of a base model (DT, RF, SVM, KNN) on different subsets of the training data. These subsets are created by randomly sampling the training data with replacements (bootstrap samples). Once all the models are trained, their predictions are combined through averaging or voting to produce the final prediction. Bagging helps reduce overfitting and variance in the model by leveraging the diversity of the trained models [103]. One popular example of bagging is RF, which constructs multiple DTs trained on random subsets of the data and aggregates their predictions [67].

Boosting is an ML ensemble method used to improve the performance of weak learners (classifiers or regressors) and convert them into strong learners [105]. It works by sequentially training multiple models, where each subsequent model focuses on the examples that were misclassified by the previous ones. Through this process, misclassified instances are assigned higher weights to prioritize their inclusion in subsequent training sets, resulting

in individual predictors specializing in different regions of the dataset [106]. In this way, boosting algorithms aim to reduce bias and variance. Some popular boosting algorithms include AdaBoost, Gradient Boosting Machines, and XGBoost.

(1) XGBoost is a modern tree classifier that enhances gradient boosting with optimizations for speed and scalability, allowing it to efficiently handle large-scale datasets [107].
(2) Gradient Boosting Machines (GBM) iteratively build a sequence of DTs, each correcting errors made by previous trees [108]. GBM demonstrates strong performance, but it faces challenges such as overfitting and computational speed [109].
(3) AdaBoost combines weak learners to create a strong learner, giving more weight to misclassified examples [110].

Stacking, also known as stacked generalization, is an ensemble learning technique that involves training multiple models and combining their predictions to make a final prediction. In stacking, the predictions made by each base model are used as features to train a meta-model, which learns how to best combine these predictions to make the final prediction. This meta-model is often a simple linear model or another ML algorithm [106,107].

Among the powerful ensemble learning techniques available, the Voting Classifier is a simple ensemble learning method that combines the predictions of multiple base classifiers (e.g., logistic regression, SVMs, DTs) and predicts the class with the most votes [111,112].

These ensemble learning models are widely used in various ML tasks such as classification, regression, and anomaly detection, and they often achieve higher predictive performance compared to individual models.

Table 2 presents a comparison of various ML models.

Table 2. Comparison between different ML models.

Learning Type	Definition	Task Type	Applications	Advantages	Disadvantages
Supervised Learning	Builds a mathematical model with a labeled training dataset that consists of both inputs and their known outputs	Classification, Regression	Spam detection, Speech recognition, object recognition	Accurate predictions for known applications, fast convergence, can handle multi-class classification	Requires labeled data which can be expensive and time-consuming, unable to identify unknown applications, requires large number of data, can suffer from overfitting when the model is too complex
Unsupervised Learning	Learning from unlabeled data without predefined output	Clustering, data aggregation	Anomaly detection	Automatically discover patterns within unlabeled datasets, discover hidden patterns and structures, Can handle large datasets	May produce ambiguous/difficult results interpretation, difficult to evaluate performance, the built clusters must still be mapped to their applications, and the number of clusters is much greater than the number of applications, which can pose a challenge for TC tasks

Table 2. *Cont.*

Learning Type	Definition	Task Type	Applications	Advantages	Disadvantages
Semi-Supervised Learning	Learning from a combination of labeled and unlabeled data	Classification, regression, clustering	Speech recognition	Utilize both labeled and unlabeled samples, can reduce the cost of labeling data	SSL means performance is inconsistent and needs enhancement, performance may depend on the quality of labeled data, to be effective may require a larger amount of unlabeled data
Reinforcement Learning	Learning through trial and error that involves rewarding desired behaviors and/or penalizing undesired ones	Decision making	Gaming	Can learn complex decision-making	Slow convergence, Requires a reward function, which may be difficult to design

4. SDN Traffic Classification Using ML

In [113], TC within an SDN/cloud environment was investigated through the application of SL. Four distinct algorithms (SVM, NB, RF, and J48 tree (C4.5)) were employed, utilizing two sets of features: features collected from observed data and default features generated from Netmate. The results for collected features indicate accuracy rates of 79.49% (SVM), 82.05% (NB), 97.44% (RF), and 82.05% (J48 tree (C4.5)), while for the generated dataset, the accuracy became 85.29% (SVM), 84.87% (NB), 95.8% (RF), and 92.86% (J48 tree (C4.5)).

Detecting and classifying conflicting flows in SDNs were discussed in [64] based on some features (action, protocol, MAC address, and IP address) using various ML algorithms (DT, SVM, EFDT, and Hybrid (DT-SVM)), and EFDT and hybrid DT-SVM algorithms were designed based on DT and SVM algorithms to achieve higher performance. The studies were carried out on two network topologies (simple tree and fat tree) with flow volumes ranging from 1000 to 100,000. The results demonstrate that EFDT has the highest accuracy.

In [114], the authors proposed a model that integrates SDN and ML algorithms for TC. SL algorithms (SVM, NB, and Nearest Centroid) were used, and the results show that the supervised models used have an accuracy of more than 90%.

In [63], it has been focused on examining and creating a TC solution using ML that could be integrated into an SDN platform. The research presented an ML-driven TC solution for SDN, leveraging existing network statistics and an offline procedure to understand network traffic patterns with the aid of a clustering algorithm. Instead of predefining a fixed number of network traffic classes, an unsupervised learning (USL) algorithm was employed to determine the most suitable number of network traffic classes, thereby offering a more customized TC approach for network operators. To accomplish this, the dataset was initially clustered and annotated using an unsupervised ML algorithm, followed by training multiple classification models based on the resulting dataset.

In Table 3, we thoroughly examine the aforementioned related works and offer a detailed comparison with respect to objective, classification models, features, dataset (topology), controller, and accuracy achieved.

Table 3. Summary of common classification ML models for SDN.

Ref.	Year	Objective	Classification Models	Required Features	Dataset	Topology Used	Controller	Accuracy
[113]	2023	Addresses TC in an SDN/cloud platform	SVM, NB, RF, and J48 tree (C4.5)	Nine features were used for the first iteration, but for the second iteration, a set of default features provided by Netmate tool were used	manually collected using the tcpdump tool	–	OpenDaylight controller	Accuracy rates are up to 97% with the studied features and exceeding 95% with the generated features
[64]	2021	Detection and classification of conflict flows in SDN	SVM, DT, Hybrid DT-SVM, Extremely Fast Decision Tree (EFDT)	Action, Protocol, MAC address, and IP	Dataset From their previous work	Fat Tree and Simple Tree Topologies	Ryu controller	98.53%, 99.27%, 99.27%; 99.49%
[114]	2020	Classification of data traffic based on the applications in an SDN platform	SVM, Nearest centroid, Naïve Bayes (NB)	Generated using the "netmate" flow generator	Generated using the "netmate" flow generator	Single switch with "n" host topology, linear topology with the same number of switches as the host, tree topology.	POX controller	92.3%, 91.02%; 96.79%
[63]	2020	Analyzing network data and deploying an ML-based network TC solution, followed by the integration of the model within an SDN platform	SVM (RBF), SVM (Linear), DT, RF, KNN	MAC addresses of source and destination, port addresses, flow duration, flow byte count, flow packet count, and average packet size	IP Network Traffic Flows dataset, labeled with 75 Apps from Kaggle	Tree Topology is used for Simulation Testbed	Ryu controller	70.40%, 96.37%, 95.76%, 94.92%; 71.47%
[115]	2020	Data Classification	RF, LDA, DNN are tested for two Scenarios	time, bytesReceived, bytesSent, durationSec, packetRx-Dropped, packetsRxError, packetsSent, packetsTx-Dropped, packetsTxError, packetsRe-ceived, rx throughput, tx throughput	Used their own dataset	Single OVS switch with two sender nodes and a receiver	ONOS controller	Scenario A: RF 95%, LDA 98% and DNN 69%, Scenario B: RF 42%, LDA 76% and DNN 74%
[116]	2017	Traffic Classification	SVM, K-Means	30 features were used	Dataset from [117]	–	–	SVM has higher accuracy than K-Means
[118]	2016	Classify traffic for SDNs in a QoS-aware manner	Laplacian SVM, KMeans	10 features	Real internet dataset captured by the Broadband Communica-tion Research group in UPC, Barcelona, Spain	–	–	Laplacian SVM approximately ranges from 80 to 90%

Table 3. *Cont.*

Ref.	Year	Objective	Classification Models	Required Features	Dataset	Topology Used	Controller	Accuracy
[42]	2016	Classify traffic for SDNs in an application-aware manner	RF, Stochastic Gradient Boosting (SGB), and Extreme Gradient Boosting (EGB)	Size, interarrival time, and timestamps of first 5 packets. Src/Dst MAC and IP addresses. Src/Dst ports. The duration, packet, and byte counts.	Labeled dataset using their own topology created an SDN application designed to collect OpenFlow statistics from the controlled switches.	–	–	RF and EGB has the highest accuracy
[119]	2014	Application-layer Classification	DPI, ML, and Multi-classifier (combination between DPI and ML)	–	Five datasets collected from their own topology	Single switch with two hosts topology	Floodlight controller	MultiClassifier has high accuracy

In [115], the authors applied various ML algorithms to classify real network traffic data automatically. To assess the performance of these algorithms on actual physical and virtual networks, two different scenarios were implemented. The first scenario involves regular data delivery over the network, while the second scenario simulates a malicious network, where the receiver node is periodically flooded with excessive requests. Results show that the second scenario has an overall lower accuracy than the first scenario.

The work performed in [116] examined two ML algorithms (SVM and K-means) for TC. The dataset used is from [117]. The results show that the overall accuracy achieved is greater than 95%.

In [118], a QoS-aware TC system was proposed that combines DPI and semi-supervised ML algorithms. DPI labels certain traffic flows that belong to known applications. The labeled data are subsequently employed by a SSL algorithm comprising Laplacian SVM and K-Means to categorize traffic flows from unknown applications. By doing so, the system can classify both known and unknown traffic flows into distinct QoS classes. Simulation results show that Laplacian SVM accuracy ranges from approximately 80% to 90%.

In [42], an application-aware TC system 2qw introduced. SDN topology is implemented to gather traffic data. Following that, multiple SL algorithms are applied to categorize traffic flows into different applications.

The work performed in [119], proposed a MultiClassifier system that identifies applications through the integration of an ML-based classifier and a DPI-based classifier. When a new flow arrives, the ML-based classifier is first used for classification. If the reliability of its classification result exceeds a predetermined threshold value, it is considered the final result of the MultiClassifier system. However, if the ML-based classifier's result's reliability is beneath the threshold, the system will resort to DPI-based classification. If the DPI-based classification returns "UNKNOWN", the classification results from the ML-based classifier will still be selected. Otherwise, the classification results from the DPI-based classifier will be selected.

From Table 3 it can be seen that the collective findings from the reviewed papers underscore the significant impact and versatility of ML techniques in the domain of TC within SDNs. The integration of SVM and DT in [64] is motivated by several reasons. One primary advantage is that DTs excel at capturing complex decision boundaries, while SVMs are adept at handling high-dimensional spaces. By combining these strengths, the hybrid model can better accommodate diverse datasets, capturing both linear and non-linear relationships effectively. Additionally, the hybrid model offers robustness to noise, drawing on SVM's noise tolerance while still leveraging DTs to discern intricate patterns. The interpretability of the model is enhanced, as DTs inherently provide clear rules for decision making, contributing to a more understandable and interpretable model.

Moreover, the hybrid model can exploit the non-linear capabilities of both SVM and decision trees, proving advantageous in scenarios where intricate relationships need to be captured. The combination also enables insights into feature importance, a benefit derived from the inherent property of DTs. The ensemble effect, derived from combining SVM and DTs, is another notable advantage, often leading to improved model performance. Additionally, the hybrid model can handle imbalanced data effectively, benefiting from decision trees' ability to address such scenarios. Lastly, the computational efficiency of the hybrid model is enhanced, with DTs being less computationally intensive compared to certain SVM configurations. Overall, the adoption of the hybrid SVM and DT model is driven by a strategic amalgamation of these advantages to address the specific requirements of the research problem at hand. In [114], the showcase emphasized the applicability of diverse ML algorithms, revealing varying performance across scenarios, while [63] uses SVM with both linear and Radial Basis Function (RBF) kernels. The observed outcomes reveal a notable performance discrepancy between the two kernels. The decision to employ the linear SVM kernel may stem from the dataset's characteristics, where the underlying relationships between features and the target variable are more effectively captured by a linear decision boundary.

Linear SVMs are particularly potent when dealing with linearly separable data, and the high accuracy achieved with this kernel in this paper underscores its appropriateness for the given context. On the other hand, the observed low accuracy with the RBF kernel suggests that the inherent flexibility and capacity to capture non-linear relationships might not be beneficial for this specific dataset. The RBF kernel introduces additional complexity, and in situations where a simpler model suffices, it may lead to overfitting or suboptimal performance. The choice between linear and RBF kernels often hinges on the characteristics of the data, and the results highlight the significance of this consideration in determining the most suitable kernel for the given research context. The achievement in [116] demonstrated impressive accuracy using SVM and K means. In [118], the proposal of a QoS-aware TC system combining DPI and semi-supervised ML algorithms demonstrated the successful categorization of known and unknown traffic flows. The application-aware TC system in [42], leveraging SDN topology, and the MultiClassifier system in [119], combining ML-based and DPI-based classifiers, further contribute to the diversity of ML approaches. In summary, putting all these studies together shows that using ML is really effective for sorting out different types of traffic in SDNs. However, it also suggests that we need to keep looking into it and make it better to deal with specific problems and work well in real-world networks.

5. Security in SDN Using ML

ML algorithms play an important role in security and TC by analyzing network traffic patterns to discern normal behavior from potential security threats. By leveraging ML for TC, SDN can precisely identify and categorize various types of network traffic, enabling targeted security measures. The integration of ML-driven TC with security protocols ensures a dynamic defense mechanism against evolving threats, as the network can adapt in real time to anomalies. This seamless collaboration between security and TC in SDN not only enhances threat-detection and -response capabilities but also contributes to the overall robustness and reliability of modern network architectures.

The implementation of a threat-aware system, known as Eunoia, as proposed by [62], utilizes ML to counter network intrusion in SDN. Initially, the data preprocessing subsystem employs a forward feature selection strategy to choose relevant feature sets. Subsequently, the predictive data modeling subsystem utilizes DT and RF algorithms to identify malicious activities. A dataset of 30,000 entries was randomly selected from 10% of the KDD99 intrusion-detection dataset based on the 1998 DARPA initiative. Results demonstrate that RF achieves an accuracy of 98.75% when using the entire dataset, 99.4% when excluding ambiguous data, and 45% when only ambiguous data are selected. Meanwhile, accuracy

for DT was measured using ambiguous data only for different numbers of features, yielding 82.48% and 91.17% for the selection of 10 and 15 features, respectively.

The data presented in Table 4 highlight the substantial influence and flexibility of ML methods in the field of TC for security in SDNs, as indicated by the collective results of the reviewed papers. In [120], ML techniques to counteract Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks in SDN are proposed and assessed. The evaluation of these techniques takes place in a realistic scenario where the SDN controller is exposed to DDoS attacks, with the aim of deriving crucial insights to enhance the security of future communication networks through ML-based approaches. The ML techniques utilized include SVM, NB, DT, and Logistic Regression, with corresponding accuracy rates of 97.5%, 96.03%, 96.78%, and 89.98%, respectively.

Table 4. Summary of common classification ML models for Security in SDN.

Ref.	Year	Objective	Classification Models	Required Features	Dataset	Accuracy
[120]	2020	Propose and evaluate the use of ML techniques to address Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks in SDN environments	SVM, NB, DT, and Logistic Regression	-	Created with the traffic flow generated by Scapy	97.5%, 96.03%, 9 6.78%; 89.98%
[62]	2017	Implementation of a threat-aware system, known as Eunoia	DT, RF	Data preprocessing subsystem employs a forward feature-selection strategy to choose relevant feature sets.	Intrusion-detection dataset based on the 1998 DARPA initiative	RF achieves accuracies of 98.75% (full dataset), 99.4% (excluding ambiguous data), and 45% (only ambiguous data), DT accuracy was assessed using ambiguous data with 10 and 15 features, resulting in 82.48% and 91.17%, respectively
[121]	2016	Investigated DDoS attacks by analyzing traffic flow patterns	NB, KNN, K-means, and K-medoids	Request time, source host, destination host, and flag bit (referred to as "fg") represent the connection request time, the originating host, the target host, and the flag bit, respectively	Real-time dataset, which is a traced file obtained from TCP traffic between Lawrence Berkley Laboratory and the rest of the world	94%, 90%, 86%; 88%
[122]	2016	Introducing an enhanced behavior-based SVM for classifying network attacks to improve the accuracy of intrusion detection and speed up the learning process for distinguishing between normal and intrusive patterns	SVM	DT used for feature reduction: prioritizes relevant features, selecting suitable ones as input for SVM	KDD1999 dataset	Average accuracy of 97.55%

The examination of DDoS attacks, as explored in [121], involves the analysis of traffic flow patterns. The focus is on distinguishing between normal and abnormal traffic by utilizing various ML algorithms, including NB, KNN, K-means, and K-medoids. The accuracy rates for the ML methods are 94%, 90%, 86%, and 88%, respectively.

In [122], an improved behavior-based SVM is introduced for the classification of network attacks. To enhance the accuracy of intrusion detection and accelerate the learning of normal and intrusive patterns, DT is employed as a feature-reduction technique. This

involves prioritizing relevant features and selecting the most qualified ones, which are then utilized as input data for training the SVM classifier. The results demonstrate an average accuracy of 97.55%.

From Table 4, it is evident that the papers reviewed present various approaches and techniques for utilizing ML in countering network intrusion in SDN environments. The study proposing the threat-aware system Eunoia [62] utilizes ML, specifically DT, and RF, to identify malicious activities in SDN. The results demonstrate high accuracy rates for RF, particularly when excluding ambiguous data. However, the accuracy significantly decreases when only ambiguous data are considered, highlighting the importance of data preprocessing and feature selection in enhancing model performance. The findings from the study outlined in [120] underscore the effectiveness of employing ML techniques to combat DoS and DDoS attacks in SDN environments. Through the utilization of SVM, NB, DT, and Logistic Regression, the study achieved notable accuracy rates, ranging from 89.98% to 97.5%. These results highlight the potential of ML-based approaches to significantly enhance the security posture of future communication networks, offering robust defenses against malicious cyber threats such as DDoS attacks. The achievement in [122] introduces an enhanced behavior-based SVM for classifying network attacks. By leveraging DT as a feature-reduction technique, the model prioritizes relevant features to enhance intrusion-detection accuracy and expedite the learning process for normal and intrusive patterns. The findings reveal an impressive average accuracy of 97.55%, showcasing the efficacy of the proposed SVM approach in accurately identifying and classifying network attacks.

In conclusion, the reviewed papers collectively demonstrate the effectiveness of ML techniques, particularly ensemble methods like RF and SVM, in detecting and mitigating network intrusion in SDN environments. Additionally, the importance of data preprocessing, feature selection, and model optimization is emphasized in improving the accuracy and robustness of ML-based intrusion detection systems.

6. Datasets

This section presents a concise overview of several recently used datasets that are valuable resources for researchers and practitioners in the field of security and network analysis using ML. These datasets encompass diverse characteristics such as benign traffic, common attacks, and network flow analysis results. Table 5 summarizes key attributes of each dataset, aiding researchers in selecting appropriate datasets for their specific needs and analyses.

Table 5. Summary of some publicly available datasets.

Ref.	Dataset Name	Year	Description	No. of Features	Created by
[123]	DDOS attack SDN Dataset	2020	This dataset is specifically designed for SDN applications, created using the Mininet emulator, and employed for TC using ML and deep learning algorithms	23 features	Mendley data
[124]	A Novel SDN Intrusion Dataset	2020	SDN dataset tailored to attacks, openly accessible to researchers, providing a comprehensive resource for evaluating intrusion detection system performance	More than 80 features	Authors of [124]
[125]	CSE-CIC-IDS2018 on AWS	2018	The project aims to create a varied and extensive benchmark dataset for intrusion detection. This involves building user profiles representing network events and behaviors, which are then combined to produce diverse datasets covering different evaluation aspects	80 features	A joint project involving the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC)

Table 5. *Cont.*

Ref.	Dataset Name	Year	Description	No. of Features	Created by
[126]	IP Network Traffic Flows Labeled with 75 Apps	2017	This dataset extends its capability by creating ML models that can identify specific applications, including Facebook, YouTube, and Instagram, among others, using IP flow statistics (currently covering 75 applications)	87 features	Universidad Del Cauca, Popayán, Colombia
[127]	CIC-IDS2017	2017	The dataset comprises benign traffic and recent common attacks, mirroring real-world PCAP data. It incorporates network traffic analysis outcomes from CICFlowMeter, featuring labeled flows categorized by timestamps, source and destination IPs, ports, protocols, and attack types in CSV format	More than 80 features	Canadian Institute for Cybersecurity (CIC)

7. Limitations and Open Research Issues

Emerging technologies, such as SDN and ML, have the potential to greatly enhance network automation and management. However, there are several limitations associated with the integration of SDN and ML. Addressing these limitations will be crucial to realizing the full potential of these technologies and achieving more efficient network management.

7.1. Limitations

7.1.1. Datasets Availability

The availability of high-quality datasets is a critical factor in the development and evaluation of ML algorithms. However, the lack of openly accessible and standardized datasets poses a significant challenge, not only in the field of SDN but also in various domains [33,128].

To address this challenge, researchers have proposed different methods for generating the necessary datasets to assess various ML algorithms in the context of SDN. One approach is the implementation of testbeds, which involve setting up real-world network environments specifically designed for data collection and experimentation [129]. Testbeds provide researchers with the flexibility to control network parameters and collect data under specific conditions, allowing for the generation of customized datasets that reflect real-world network characteristics.

Another approach is the use of standard network simulators or emulators, such as Mininet, NS3, or EstiNet [130–133]. These simulators and emulators provide virtualized environments where network behaviors can be simulated, allowing researchers to generate datasets that capture various network scenarios and conditions. Simulators and emulators enable reproducibility and scalability, as experiments can be easily replicated and expanded upon.

While these approaches offer valuable means of generating datasets, it is important to note that they have their limitations. Testbeds may require significant resources and infrastructure, making them costly and challenging to set up. Simulators and emulators, on the other hand, may introduce certain simplifications and assumptions that do not perfectly reflect the complexities of real-world network environments.

7.1.2. Datasets Quality

The quality and availability of datasets play a critical role in the training and performance of ML models, and this holds true in the context of SDN environments as well.

However, there are specific challenges associated with dataset quality in SDN that need to be addressed for effective ML-based solutions [134,135].

One of the primary challenges is the scarcity of labeled data in SDN environments. ML models typically require large numbers of accurately labeled data for training. However, in the context of SDN, acquiring such labeled datasets can be challenging due to the complexity and dynamic nature of network traffic. The manual labeling of data is time-consuming, expensive, and prone to errors. Additionally, the diversity and scale of network traffic in SDN make it difficult to gather representative and comprehensive datasets that capture all relevant traffic patterns.

To overcome the challenge of limited labeled data, researchers and practitioners in SDN have explored various approaches. One approach is to leverage transfer learning, where models trained on related datasets or domains are fine-tuned or used as a starting point for training in the target SDN environment. This allows for the transfer of knowledge and experiences from existing labeled datasets to the target scenario, reducing the reliance on scarce labeled data.

7.1.3. High-Bandwidth Traffic Classification

A significant challenge for TC systems is the significant progress in the network, which may need to process traffic at gigabit speeds in some instances. Ref. [43] proposed two solutions to address this challenge:

1. Utilizing specialized hardware or parallel processing architecture.
2. To enhance the scalability of SDN classification, it is recommended that the architecture of SDN be restructured. Relevant studies have suggested practical techniques [136,137].

7.1.4. Interpretability and Clarity

One of the challenges associated with many ML algorithms, especially those based on deep learning, is their inherent lack of interpretability, often referred to as the "black box" nature of these models [138]. This lack of interpretability makes it difficult to comprehend and explain the decision-making processes of the models, hindering their transparency and understandability.

In the context of TC in SDN, interpretability and clarity are crucial, particularly in scenarios where transparency is essential, such as network security. Understanding why and how a particular traffic flow is classified as belonging to a specific class becomes crucial for network administrators and security analysts to make informed decisions and take appropriate actions.

The ability to interpret ML models' decisions can provide valuable insights into the reasoning behind TC outcomes. It allows network operators to understand the factors and features that contribute to the classification decision, enabling them to validate the accuracy of the classifications and gain confidence in the model's performance. Interpretability also facilitates the identification of potential biases or shortcomings in the model's training data or architecture, allowing for improvements and adjustments to be made.

Addressing the challenge of interpretability and clarity in ML-based TC requires the development of techniques and methodologies that can provide meaningful explanations for the model's decisions. This involves exploring approaches such as model-agnostic interpretability techniques, rule-extraction methods, and feature importance analysis. By leveraging these techniques, it becomes possible to extract interpretable rules or explanations from complex ML models, shedding light on the factors influencing the classification outcomes.

7.1.5. Ideal Network Assumption

Many existing research studies in the field of ML-based TC and SDN assume ideal network conditions where complete and accurate information about traffic flow is readily available. However, in reality, real-world networks often encounter various anomalies and challenges that can significantly impact network performance and the effectiveness of

ML-based approaches. These anomalies include packet loss, packet retransmission, delay, and jitter, which can lead to deviations from the expected traffic patterns and introduce uncertainties in the classification process.

The presence of these abnormal conditions poses significant challenges to the efficiency and accuracy of ML-based TC models. ML algorithms trained on ideal network assumptions may struggle to handle the complexities and variations introduced by real-world network conditions. As a result, classification accuracy may suffer, and the reliability of the TC system may be compromised.

To address this issue, it is crucial to develop robust traffic classifiers that can effectively handle and adapt to abnormal network conditions. These classifiers should be designed to be resilient to packet loss, retransmission, delays, and jitter, ensuring accurate classification even in the presence of such anomalies. Additionally, techniques such as anomaly detection and outlier handling can be incorporated into the ML algorithms to identify and mitigate the impact of abnormal network behavior on the classification process.

7.1.6. Resources Limitations

ML algorithms may require significant computational resources. In SDN environments with limited resources, this may be a constraining factor [139].

7.2. Open Research Issues

7.2.1. Real-World Challenges

While much of the existing research in this field has focused on simulating ML algorithms on simple network topologies and assessing the accuracy of the models, it is essential to acknowledge that real-world networks present significantly greater complexity. Merely achieving high accuracy in a controlled environment is insufficient when it comes to practical implementation. Real-world network performance is influenced by a multitude of factors, such as scalability, availability, and adaptability to dynamic conditions.

In order to address the challenges of real-world network environments, it is imperative to consider the scalability of ML-based solutions. As network sizes and traffic volumes increase, the algorithms must be able to handle the corresponding growth without compromising efficiency or accuracy. Additionally, the availability of ML models is crucial, as the network must continue to operate reliably even in the face of failures or disruptions.

Another critical aspect to consider is the utilization of larger and more diverse datasets. While many studies have demonstrated the effectiveness of ML algorithms using relatively small and homogeneous datasets, real-world networks exhibit a wide range of traffic patterns, protocols, and applications. By incorporating more comprehensive datasets that capture this diversity, ML models can be trained to better handle the complexities and idiosyncrasies of real-world traffic.

Furthermore, it is important to consider the practical implementation challenges associated with integrating ML algorithms into existing network infrastructures. Network operators must navigate issues related to the deployment, management, and maintenance of ML models in a live network environment. These challenges include issues such as computational requirements, model updates, and integration with existing network management systems.

To overcome these real-world challenges, future research should focus on developing ML algorithms that are specifically designed for complex network topologies and can effectively address scalability, availability, and adaptability concerns. Furthermore, efforts should be made to collect and analyze larger and more diverse datasets that accurately represent real-world network traffic. Only by addressing these challenges can ML-based solutions be successfully deployed and utilized in practical network environments, leading to improved network performance and enhanced QoS.

7.2.2. Architecture Generalization

In traditional networks, communication between non-adjacent layers is typically restricted, limiting the potential for information exchange and collaboration across different network domains [140]. However, enabling cross-domain generalization is crucial when applying ML models trained on one specific SDN architecture to diverse network types or architectures.

To achieve effective generalization, it is necessary to conduct research and develop approaches that can seamlessly adapt ML models to various network architectures. This entails designing models that can capture the underlying principles and patterns common to different SDN architectures, enabling the transfer of knowledge and experiences gained from one architecture to another. By doing so, ML models trained on a specific SDN architecture can be effectively applied to different network environments, reducing the need for extensive retraining or model redesign.

Furthermore, it is important to explore techniques that facilitate the transfer of learned knowledge from one SDN architecture to another. This includes investigating methods for extracting and abstracting architectural-agnostic features and representations that capture the essential characteristics of network behavior. By focusing on these architecture-agnostic features, ML models can better adapt to diverse network architectures, allowing for more efficient and generalized deployment.

Additionally, the development of standardized interfaces and protocols across different SDN architectures can greatly facilitate architecture generalization. By establishing common standards for communication and information exchange between various layers and domains, ML models can seamlessly integrate with different architectures, promoting interoperability and flexibility.

Overall, conducting research on approaches that enable cross-domain generalization in ML-based SDN applications is crucial for advancing the practicality and scalability of these technologies. By developing models and techniques that can effectively adapt to diverse network architectures, we can unlock the full potential of ML in SDN and reap the benefits of improved network performance, enhanced resource utilization, and enhanced QoS across a wide range of network environments.

7.2.3. Use of Formal Methods and Model-Based Testing

Formal methods and model-based testing play a crucial role in the context of ML-based TC techniques in SDNs [141]. Formal methods provide a rigorous framework for specifying and verifying the properties of network protocols and algorithms, enabling the detection of potential vulnerabilities or design flaws in SDN-based TC systems, and ensuring their correctness and reliability [142]. By employing formal methods, researchers and practitioners can mathematically analyze the behavior and performance of the ML models and algorithms used for TC. This analysis helps in identifying potential limitations, biases, or vulnerabilities in the models and enables the development of robust and accurate TC solutions [143]. Additionally, model-based testing techniques allow for systematic and automated testing of the TC algorithms against well-defined models or specifications, helping in validating the behavior and performance of the algorithms under different traffic scenarios, enhancing their effectiveness, and ensuring their suitability for real-world deployment in SDN environments [144]. Overall, the use of formal methods and model-based testing contributes to the reliability, accuracy, and efficiency of ML-based TC techniques in SDNs.

7.2.4. Use of Ensemble Learning Models

Ensemble models, which combine the predictions of multiple individual models, have emerged as powerful tools in ML for improving predictive accuracy and robustness [103,109]. Ensemble learning, despite being a potent tool in ML, poses numerous research challenges that demand deeper exploration. One critical concern concentrates on the scalability and efficiency of ensemble methods, especially in dealing with large-scale datasets and real-

time applications. With datasets continually expanding in size and complexity, there is a pressing need to devise ensemble learning strategies capable of efficiently managing such extensive data volumes without sacrificing predictive accuracy [145]. Moreover, the interpretability and transparency of ensemble models pose significant challenges. This is particularly evident in ensemble methods where multiple base learners are combined, each with distinct parameters and decision-making approaches. Enhancing the interpretability of ensemble models is essential for extracting insights into the underlying data relationships and instilling confidence in the model's predictions [146]. Addressing these open research issues in ensemble learning will not only advance the field but also enhance the applicability and robustness of ensemble methods across various domains and applications.

7.2.5. Routing Optimization and Resource Management

Routing optimization and resource management in SDN present intriguing avenues for exploration, particularly in the context of leveraging ML techniques. One of the open research issues in this domain is the development of ML-based algorithms for routing optimization. Traditional routing protocols within SDN might not fully exploit the dynamic nature of network traffic and changes in network topology. ML algorithms can adaptively learn from network data to optimize routing decisions, leading to improved network performance, reduced latency, and enhanced QoS [147,148].

Resource management poses another challenge, as efficiently allocating network resources based on varying demand patterns is crucial for maintaining network reliability and efficiency. ML models can analyze historical traffic patterns and resource usage data to predict future demand and dynamically adjust resource allocations accordingly [149].

Overall, delving into ML applications for optimizing routing and managing resources in SDN presents an exciting area for future research and innovation, with the potential to significantly enhance the performance and scalability of modern networks.

8. Conclusions

SDN and ML are innovative technologies that have the potential to greatly enhance network performance and QoS. SDN facilitates centralized and programmable network management, enabling efficient resource utilization and dynamic adaptation to changing traffic demands. ML, on the other hand, can analyze network data to identify patterns and forecast future traffic behavior, offering proactive QoS management capabilities. When combined with TC, SDN and ML can accurately identify and prioritize different traffic types, optimizing network performance, mitigating congestion, and improving the overall user experience.

However, the effectiveness of this approach heavily relies on the quality and quantity of data used for analysis. By leveraging larger and more diverse datasets, the accuracy and robustness of these technologies can be significantly enhanced, unlocking their full potential in improving network performance and QoS management. Therefore, future research should focus on collecting and utilizing comprehensive datasets to further advance the application of ML algorithms in the context of SDN.

This paper provided a comprehensive survey of the application of ML algorithms in the domain of SDN, with a specific emphasis on TC. We discussed the differences between traditional and ML-based TC methods, highlighting the advantages offered by ML techniques. Additionally, we provided an overview of various ML algorithms that have been applied in SDN environments. By examining the existing literature, we explored the current state of the field and identified key research limitations and open issues that require further investigation.

Despite the progress made, there are still several challenges that need to be addressed in the field of ML and SDNs. Collaboration among researchers is crucial in overcoming these challenges and advancing the field. By working together, we can make new discoveries and develop innovative approaches that will shape the future of traffic categorization in

SDNs. This survey serves as a valuable reference, providing insights into the current state of the field and inspiring further exploration in this rapidly evolving area.

Author Contributions: Conceptualization, R.H.S., M.S.A., H.A.E.A.E., M.S. and M.M.S.; Methodology, R.H.S., M.S.A., H.A.E.A.E. and M.K.; Formal analysis, R.H.S., M.S.A., M.K. and M.M.S.; Investigation, R.H.S., M.S.A., H.A.E.A.E. and M.K.; Resources, R.H.S., M.S.A., M.S., M.K. and M.M.S.; Data curation, R.H.S. and M.S.A.; Writing—original draft, R.H.S., M.S.A., H.A.E.A.E., M.S., M.K. and M.M.S.; Writing—review & editing, R.H.S., M.S.A., H.A.E.A.E., M.S., M.K. and M.M.S.; Visualization, R.H.S., M.S.A., M.K. and M.M.S.; Supervision, M.S.A., H.A.E.A.E. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

A-CPI	Application-Controller Plan Interface
AI	Artificial Intelligence
API	Application Programming Interface
C-DPI	Control-Data Plane Interface
DPI	Deep Packet Inspection
DT	Decision Tree
EFDT	Extremely Fast Decision Tree
IANA	Internet Assigned Numbers Authority
ISP	Internet Service Provider
KNN	Key Nearest Neighbor
ML	Machine Learning
NB	Naïve Bayes
QoS	Quality of Service
RBF	Radial Basis Function
RF	Random Forest
SDN	Software Defined Network
SL	Supervised Learning
SOM	Self-Organizing Map
SSL	Semi-Supervised Learning
SVM	Support Vector Machine
TC	Traffic Classification
USL	Unsupervised Learning

References

1. Latah, M.; Toker, L. Application of artificial intelligence to software defined networking: A survey. *Indian J. Sci. Technol.* **2016**, *9*, 1–7. [CrossRef]
2. Sezer, S.; Scott-Hayward, S.; Chouhan, P.K.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun. Mag.* **2013**, *51*, 36–43. [CrossRef]
3. Rowshanrad, S.; Namvarasl, S.; Abdi, V.; Hajizadeh, M.; Keshtgary, M. A survey on SDN, the future of networking. *J. Adv. Comput. Sci. Technol.* **2014**, *3*, 232–248. [CrossRef]
4. Huo, L.; Jiang, D.; Qi, S.; Miao, L. A blockchain-based security traffic measurement approach to software defined networking. *Mob. Netw. Appl.* **2021**, *26*, 586–596. [CrossRef]
5. Wang, Y.; Jiang, D.; Huo, L.; Zhao, Y. A new traffic prediction algorithm to software defined networking. *Mob. Netw. Appl.* **2021**, *26*, 716–725. [CrossRef]
6. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
7. Farhady, H.; Lee, H.; Nakao, A. Software-defined networking: A survey. *Comput. Netw.* **2015**, *81*, 79–95. [CrossRef]
8. Shu, Z.; Wan, J.; Li, D.; Lin, J.; Vasilakos, A.V.; Imran, M. Security in software-defined networking: Threats and countermeasures. *Mob. Netw. Appl.* **2016**, *21*, 764–776. [CrossRef]

9. Karakus, M.; Durresi, A. Quality of service (QoS) in software defined networking (SDN): A survey. *J. Netw. Comput. Appl.* **2017**, *80*, 200–218. [CrossRef]
10. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX: Towards an operating system for networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 105–110. [CrossRef]
11. Gupta, N.; Maashi, M.S.; Tanwar, S.; Badotra, S.; Aljebreen, M.; Bharany, S. A comparative study of software defined networking controllers using mininet. *Electronics* **2022**, *11*, 2715. [CrossRef]
12. Kaur, S.; Singh, J.; Ghumman, N.S. Network programmability using POX controller. In Proceedings of the ICCCS International Conference on Communication, Computing & Systems, Chennai, India, 20–21 February 2014; Volume 138, p. 70.
13. Medved, J.; Varga, R.; Tkacik, A.; Gray, K. Opendaylight: Towards a model-driven sdn controller architecture. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, Australia, 19 June 2014; pp. 1–6.
14. Albu-Salih, A.T. Performance evaluation of ryu controller in software defined networks. *J. Qadisiyah Comput. Sci. Math.* **2022**, *14*, 1. [CrossRef]
15. Erickson, D. The beacon openflow controller. In Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, 16 August 2013; pp. 13–18.
16. Blial, O.; Ben Mamoun, M.; Benaini, R. An overview on SDN architectures with multiple controllers. *J. Comput. Netw. Commun.* **2016**, *2016*, 9396525. [CrossRef]
17. Paliwal, M.; Shrimankar, D.; Tembhurne, O. Controllers in SDN: A review report. *IEEE Access* **2018**, *6*, 36256–36270. [CrossRef]
18. Tadros, C.N.; Mokhtar, B.; Rizk, M.R. Logically centralized-physically distributed software defined network controller architecture. In Proceedings of the 2018 IEEE Global Conference on Internet of Things (GCIoT), Alexandria, Egypt, 5–7 December 2018; pp. 1–5.
19. Chaudhry, S.; Bulut, E.; Yuksel, M. A Distributed SDN Application for Cross-Institution Data Access. In Proceedings of the 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019; pp. 1–9.
20. Ahmed, H.G.; Ramalakshmi, R. Performance analysis of centralized and distributed SDN controllers for load balancing application. In Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–12 May 2018; pp. 758–764.
21. Hu, T.; Guo, Z.; Yi, P.; Baker, T.; Lan, J. Multi-controller based software-defined networking: A survey. *IEEE Access* **2018**, *6*, 15980–15996. [CrossRef]
22. Xu, G.; Mu, Y.; Liu, J. Inclusion of artificial intelligence in communication networks and services. *ITU J. ICT Discov. Spec* **2017**, *1*, 1–6.
23. Zhang, J.; Guo, H.; Liu, J. Adaptive task offloading in vehicular edge computing networks: A reinforcement learning based scheme. *Mob. Netw. Appl.* **2020**, *25*, 1736–1745. [CrossRef]
24. Moustafa, S.S.; Mohamed, G.E.A.; Elhadidy, M.S.; Abdalzaher, M.S. Machine learning regression implementation for high-frequency seismic wave attenuation estimation in the Aswan Reservoir area, Egypt. *Environ. Earth Sci.* **2023**, *82*, 307. [CrossRef]
25. Hamdy, O.; Gaber, H.; Abdalzaher, M.S.; Elhadidy, M. Identifying exposure of urban area to certain seismic hazard using machine learning and GIS: A case study of greater Cairo. *Sustainability* **2022**, *14*, 10722. [CrossRef]
26. Zhao, Y.; Li, Y.; Zhang, X.; Geng, G.; Zhang, W.; Sun, Y. A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access* **2019**, *7*, 95397–95417. [CrossRef]
27. Namasudra, S.; Lorenz, P.; Ghosh, U. The New Era of Computer Network by using Machine Learning. *Mob. Netw. Appl.* **2023**, *28*, 764–766. [CrossRef]
28. Abdalzaher, M.S.; Soliman, M.S.; El-Hady, S.M.; Benslimane, A.; Elwekeil, M. A Deep Learning Model for Earthquake Parameters Observation in IoT System-Based Earthquake Early Warning. *IEEE Internet Things J.* **2022**, *9*, 8412–8424. [CrossRef]
29. Salazar, E.; Azurdia-Meza, C.A.; Zabala-Blanco, D.; Bolufé, S.; Soto, I. Semi-supervised extreme learning machine channel estimator and equalizer for vehicle to vehicle communications. *Electronics* **2021**, *10*, 968. [CrossRef]
30. Sarker, I.H. Machine learning: Algorithms, real-world applications and research directions. *SN Comput. Sci.* **2021**, *2*, 160. [CrossRef]
31. Abdalzaher, M.S.; Elwekeil, M.; Wang, T.; Zhang, S. A Deep Autoencoder Trust Model for Mitigating Jamming Attack in IoT Assisted by Cognitive Radio. *IEEE Syst. J.* **2022**, *16*, 3635–3645. [CrossRef]
32. Shafin, S.S.; Karmakar, G.; Mareels, I. Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications. *Sensors* **2023**, *23*, 5348. [CrossRef]
33. Xie, J.; Yu, F.R.; Huang, T.; Xie, R.; Liu, J.; Wang, C.; Liu, Y. A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 393–430. [CrossRef]
34. Dias, K.L.; Pongelupe, M.A.; Caminhas, W.M.; de Errico, L. An innovative approach for real-time network traffic classification. *Comput. Netw.* **2019**, *158*, 143–157. [CrossRef]
35. Owusu, A.I.; Nayak, A. An intelligent traffic classification in sdn-iot: A machine learning approach. In Proceedings of the 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Odessa, Ukraine, 26–29 May 2020; pp. 1–6.
36. Nguyen, T.T.; Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Commun. Surv. Tutor.* **2008**, *10*, 56–76. [CrossRef]

37. Thazin, N. QoS-based Traffic Engineering in Software Defined Networking. In Proceedings of the 2019 25th Asia-Pacific Conference on Communications (APCC), Ho Chi Minh City, Vietnam, 6–8 November 2019.
38. Tahaei, H.; Afifi, F.; Asemi, A.; Zaki, F.; Anuar, N.B. The rise of traffic classification in IoT networks: A survey. *J. Netw. Comput. Appl.* **2020**, *154*, 102538. [CrossRef]
39. Mohammed, A.R.; Mohammed, S.A.; Shirmohammadi, S. Machine learning and deep learning based traffic classification and prediction in software defined networking. In Proceedings of the 2019 IEEE International Symposium on Measurements & Networking (M&N), Catania, Italy, 8–10 July 2019; pp. 1–6.
40. Rojas, J.S.; Gallón, Á.R.; Corrales, J.C. Personalized service degradation policies on OTT applications based on the consumption behavior of users. In Proceedings of the Computational Science and Its Applications—ICCSA 2018: 18th International Conference, Melbourne, Australia, 2–5 July 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 543–557.
41. Fernández, D.P. Restraining ICANN: An analysis of OFAC sanctions and their impact on the Internet Corporation for Assigned Names and Numbers. *Telecommun. Policy* **2023**, *47*, 102614. [CrossRef]
42. Amaral, P.; Dinis, J.; Pinto, P.; Bernardo, L.; Tavares, J.; Mamede, H.S. Machine learning in software defined networks: Data collection and traffic classification. In Proceedings of the 2016 IEEE 24th International conference on network protocols (ICNP), Singapore, 8–11 November 2016; pp. 1–5.
43. Yan, J.; Yuan, J. A survey of traffic classification in software defined networks. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018; pp. 200–206.
44. Moore, A.W.; Papagiannaki, K. Toward the accurate identification of network applications. In Proceedings of the Passive and Active Network Measurement: 6th International Workshop, PAM 2005, Boston, MA, USA, 31 March–1 April 2005; Proceedings 6; Springer: Berlin/Heidelberg, Germany, 2005; pp. 41–54.
45. Madhukar, A.; Williamson, C. A longitudinal study of P2P traffic classification. In Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation, Monterey, CA, USA, 11–14 September 2006; pp. 179–188.
46. Finsterbusch, M.; Richter, C.; Rocha, E.; Muller, J.A.; Hanssgen, K. A survey of payload-based traffic classification approaches. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1135–1156. [CrossRef]
47. Sen, S.; Spatscheck, O.; Wang, D. Accurate, scalable in-network identification of p2p traffic using application signatures. In Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, 19–21 May 2004; pp. 512–521.
48. Goo, Y.H.; Shim, K.S.; Lee, S.K.; Kim, M.S. Payload signature structure for accurate application traffic classification. In Proceedings of the 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, Japan, 5–7 October 2016; pp. 1–4.
49. Fu, Z.; Liu, Z.; Li, J. Efficient parallelization of regular expression matching for deep inspection. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–9.
50. Gabilondo, Á.; Fernández, Z.; Viola, R.; Martín, Á.; Zorrilla, M.; Angueira, P.; Montalbán, J. Traffic classification for network slicing in mobile networks. *Electronics* **2022**, *11*, 1097. [CrossRef]
51. Guo, D.; Liao, G.; Bhuyan, L.N.; Liu, B.; Ding, J.J. A scalable multithreaded l7-filter design for multi-core servers. In Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, San Jose, CA, USA, 6–7 November 2008; pp. 60–68.
52. Gringoli, F.; Salgarelli, L.; Dusi, M.; Cascarano, N.; Risso, F.; Claffy, K. Gt: Picking up the truth from the ground for internet traffic. *ACM SIGCOMM Comput. Commun. Rev.* **2009**, *39*, 12–18.
53. Yu, C.; Lan, J.; Xie, J.; Hu, Y. QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs. *Procedia Comput. Sci.* **2018**, *131*, 1209–1216. [CrossRef]
54. Parsaei, M.R.; Sobouti, M.J.; Javidan, R. Network traffic classification using machine learning techniques over software defined networks. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*.
55. Ibrahim, H.A.H.; Al Zuobi, O.R.A.; Al-Namari, M.A.; MohamedAli, G.; Abdalla, A.A.A. Internet traffic classification using machine learning approach: Datasets validation issues. In Proceedings of the 2016 Conference of Basic Sciences and Engineering Studies (SGCAC), Khartoum, Sudan, 20–23 February 2016; pp. 158–166.
56. Audah, M.F.; Chin, T.S.; Zulfadzli, Y.; Lee, C.K.; Rizaluddin, K. Towards efficient and scalable machine learning-based QoS traffic classification in software-defined network. In Proceedings of the Mobile Web and Intelligent Information Systems: 16th International Conference, MobiWIS 2019, Istanbul, Turkey, 26–28 August 2019; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2019; pp. 217–229.
57. Dainotti, A.; Pescape, A.; Claffy, K.C. Issues and future directions in traffic classification. *IEEE Netw.* **2012**, *26*, 35–40. [CrossRef]
58. Xue, Y.; Wang, D.; Zhang, L. Traffic classification: Issues and challenges. In Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC), San Diego, CA USA, 28–31 January 2013; pp. 545–549.
59. Nikravesh, A.Y.; Ajila, S.A.; Lung, C.H. An autonomic prediction suite for cloud resource provisioning. *J. Cloud Comput.* **2017**, *6*, 1–20. [CrossRef]
60. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerg. Artif. Intell. Appl. Comput. Eng.* **2007**, *160*, 3–24.
61. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning*; Springer Series in Statistics; Springer: New York, NY, USA, 2001.

62. Song, C.; Park, Y.; Golani, K.; Kim, Y.; Bhatt, K.; Goswami, K. Machine-learning based threat-aware system in software defined networks. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–9.
63. Perera Jayasuriya Kuranage, M.; Piamrat, K.; Hamma, S. Network traffic classification using machine learning for software defined networks. In Proceedings of the Machine Learning for Networking: Second IFIP TC 6 International Conference, MLN 2019, Paris, France, 3–5 December 2019; Springer: New York, NY, USA, 2020; pp. 28–39.
64. Khairi, M.H.H.; Ariffin, S.H.S.; Latiff, N.M.A.; Yusof, K.M.; Hassan, M.K.; Al-Dhief, F.T.; Hamdan, M.; Khan, S.; Hamzah, M. Detection and classification of conflict flows in SDN using machine learning algorithms. *IEEE Access* **2021**, *9*, 76024–76037. [CrossRef]
65. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
66. Maimon, O.Z.; Rokach, L. *Data Mining with Decision Trees: Theory and Applications*; World Scientific: Singapore, 2014; Volume 81.
67. Biau, G.; Scornet, E. A random forest guided tour. *Test* **2016**, *25*, 197–227. [CrossRef]
68. Genuer, R.; Poggi, J.M.; Genuer, R.; Poggi, J.M. *Random Forests*; Springer: Berlin/Heidelberg, Germany, 2020.
69. Liaw, A.; Wiener, M. Classification and regression by randomForest. *R News* **2002**, *2*, 18–22.
70. Ridwan, M.A.; Radzi, N.A.M.; Abdullah, F.; Jalil, Y. Applications of machine learning in networking: A survey of current issues and future challenges. *IEEE Access* **2021**, *9*, 52523–52556. [CrossRef]
71. Haddouchi, M.; Berrado, A. A survey of methods and tools used for interpreting random forest. In Proceedings of the 2019 1st International Conference on Smart Systems and Data Science (ICSSD), Rabat, Morocco, 3–4 October 2019; pp. 1–6.
72. Torizuka, K.; Oi, H.; Saitoh, F.; Ishizu, S. Benefit segmentation of online customer reviews using random forest. In Proceedings of the 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, 16–19 December 2018; pp. 487–491.
73. Aria, M.; Cuccurullo, C.; Gnasso, A. A comparison among interpretative proposals for Random Forests. *Mach. Learn. Appl.* **2021**, *6*, 100094. [CrossRef]
74. Vapnik, V. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998; Volume 1, p. 2.
75. Steinwart, I.; Christmann, A. *Support Vector Machines*; Springer Science & Business Media: New York, NY, USA, 2008.
76. Martínez-Ramón, M.; Christodoulou, C. Support vector machines for antenna array processing and electromagnetics. In *Synthesis Lectures on Computational Electromagnetics*; Springer: Cham, Switzerland, 2005; Volume 1, pp. 1–120.
77. Hu, H.; Wang, Y.; Song, J. Signal classification based on spectral correlation analysis and SVM in cognitive radio. In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), Okinawa, Japan, 28–31 March 2008; pp. 883–887.
78. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
79. Box, G.E.; Tiao, G.C. *Bayesian Inference in Statistical Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
80. Bayes, T. Naive Bayes Classifier. In *Article Sources and Contributors*; BIOMISA, Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST): Islamabad, Pakistan, 1968; pp. 1–9.
81. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2020.
82. Williams, N.; Zander, S.; Armitage, G. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *ACM SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 5–16. [CrossRef]
83. Liu, J.; Xu, Q. Machine learning in software defined network. In Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019; pp. 1114–1120.
84. Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 881–892. [CrossRef]
85. Stefanovič, P.; Kurasova, O. Visual analysis of self-organizing maps. *Nonlinear Anal. Model. Control* **2011**, *16*, 488–504. [CrossRef]
86. Van Hulle, M.M. Self-organizing Maps. In *Handbook of Natural Computing*; Springer: Berlin/Heidelberg, Germany, 2012.
87. Ghaseminezhad, M.; Karami, A. A novel self-organizing map (SOM) neural network for discrete groups of data clustering. *Appl. Soft Comput.* **2011**, *11*, 3771–3778. [CrossRef]
88. Xu, L.; Xu, Y.; Chow, T.W. PolSOM: A new method for multidimensional data visualization. *Pattern Recognit.* **2010**, *43*, 1668–1675. [CrossRef]
89. Zhu, X.J. *Semi-Supervised Learning Literature Survey*; University of Wisconsin-Madison Department of Computer Sciences: Madison, WI, USA, 2005.
90. Lee, D.H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Proceedings of the Workshop on Challenges in Representation Learning, ICML, Atlanta, GA, USA, 16–21 June 2013; Volume 3, p. 896.
91. Wu, H.; Prasad, S. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Trans. Image Process.* **2017**, *27*, 1259–1270. [CrossRef] [PubMed]
92. Chapelle, O.; Scholkopf, B.; Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Trans. Neural Netw.* **2009**, *20*, 542. [CrossRef]
93. Pise, N.N.; Kulkarni, P. A survey of semi-supervised learning methods. In Proceedings of the 2008 International Conference on Computational Intelligence and Security, Suzhou, China, 13–17 December 2008; Volume 2, pp. 30–34.
94. Moon, T.K. The expectation-maximization algorithm. *IEEE Signal Process. Mag.* **1996**, *13*, 47–60. [CrossRef]

95. Ng, S.K.; Krishnan, T.; McLachlan, G.J. The EM algorithm. In *Handbook of Computational Statistics: Concepts and Methods*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 139–172.
96. Chen, Y.; Wang, G.; Dong, S. Learning with progressive transductive support vector machine. *Pattern Recognit. Lett.* **2003**, *24*, 1845–1855. [CrossRef]
97. Singla, A.; Patra, S.; Bruzzone, L. A novel classification technique based on progressive transductive SVM learning. *Pattern Recognit. Lett.* **2014**, *42*, 101–106. [CrossRef]
98. Bruzzone, L.; Chi, M.; Marconcini, M. A novel transductive SVM for semisupervised classification of remote-sensing images. *IEEE Trans. Geosci. Remote Sens.* **2006**, *44*, 3363–3373. [CrossRef]
99. Nassar, I.; Herath, S.; Abbasnejad, E.; Buntine, W.; Haffari, G. All labels are not created equal: Enhancing semi-supervision via label grouping and co-training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 17–21 June 2021; pp. 7241–7250.
100. Han, W.; Coutinho, E.; Ruan, H.; Li, H.; Schuller, B.; Yu, X.; Zhu, X. Semi-supervised active learning for sound classification in hybrid learning environments. *PLoS ONE* **2016**, *11*, e0162075. [CrossRef]
101. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, MA, USA, 1998; Volume 135.
102. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
103. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2020**, *14*, 241–258. [CrossRef]
104. Syarif, I.; Zaluska, E.; Prugel-Bennett, A.; Wills, G. Application of bagging, boosting and stacking to intrusion detection. In Proceedings of the Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, 13–20 July 2012; Proceedings 8; Springer: Berlin/Heidelberg, Germany, 2012; pp. 593–602.
105. Bartlett, P.; Freund, Y.; Lee, W.S.; Schapire, R.E. Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Stat.* **1998**, *26*, 1651–1686. [CrossRef]
106. Graczyk, M.; Lasota, T.; Trawiński, B.; Trawiński, K. Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In Proceedings of the Intelligent Information and Database Systems: Second International Conference, ACIIDS, Hue City, Vietnam, 24–26 March 2010; Proceedings, Part II 2; Springer: Berlin/Heidelberg, Germany, 2010; pp. 340–350.
107. Rashid, M.; Kamruzzaman, J.; Imam, T.; Wibowo, S.; Gordon, S. A tree-based stacking ensemble technique with feature selection for network intrusion detection. *Appl. Intell.* **2022**, *52*, 9768–9781. [CrossRef]
108. Natekin, A.; Knoll, A. Gradient boosting machines, a tutorial. *Front. Neurorobotics* **2013**, *7*, 21. [CrossRef] [PubMed]
109. Eom, W.J.; Song, Y.J.; Park, C.H.; Kim, J.K.; Kim, G.H.; Cho, Y.Z. Network traffic classification using ensemble learning in software-defined networks. In Proceedings of the 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 13–16 April 2021; pp. 89–92.
110. Lee, W.; Jun, C.H.; Lee, J.S. Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification. *Inf. Sci.* **2017**, *381*, 92–103. [CrossRef]
111. Dagnew, G.; Shekar, B. Ensemble learning-based classification of microarray cancer data on tree-based features. *Cogn. Comput. Syst.* **2021**, *3*, 48–60. [CrossRef]
112. Kim, H.; Kim, H.; Moon, H.; Ahn, H. A weight-adjusted voting algorithm for ensembles of classifiers. *J. Korean Stat. Soc.* **2011**, *40*, 437–449. [CrossRef]
113. Belkadi, O.; Vulpe, A.; Laaziz, Y.; Halunga, S. ML-Based Traffic Classification in an SDN-Enabled Cloud Environment. *Electronics* **2023**, *12*, 269. [CrossRef]
114. Raikar, M.M.; Meena, S.; Mulla, M.M.; Shetti, N.S.; Karanandi, M. Data traffic classification in software defined networks (SDN) using supervised-learning. *Procedia Comput. Sci.* **2020**, *171*, 2750–2759. [CrossRef]
115. Kwon, J.; Jung, D.; Park, H. Traffic data classification using machine learning algorithms in SDN networks. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 21–23 October 2020; pp. 1031–1033.
116. Fan, Z.; Liu, R. Investigation of machine learning based network traffic classification. In Proceedings of the 2017 International Symposium on Wireless Communication Systems (ISWCS), Bologna, Italy, 28–31 August 2017; pp. 1–6.
117. Moore, A.; Hall, J.; Kreibich, C.; Harris, E.; Pratt, I. Architecture of a network monitor. In Proceedings of the Passive & Active Measurement Workshop, San Diego, CA, USA, 6–8 April 2003 2003; Volume 2003.
118. Wang, P.; Lin, S.C.; Luo, M. A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA, 27 June–2 July 2016; pp. 760–765.
119. Li, Y.; Li, J. MultiClassifier: A combination of DPI and ML for application-layer classification in SDN. In Proceedings of the 2014 2nd International Conference on Systems and Informatics (ICSAI 2014), Shanghai, China, 15–17 November 2014; pp. 682–686.
120. Ahmad, A.; Harjula, E.; Ylianttila, M.; Ahmad, I. Evaluation of machine learning techniques for security in SDN. In Proceedings of the 2020 IEEE Globecom Workshops (GC Wkshps), Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
121. Barki, L.; Shidling, A.; Meti, N.; Narayan, D.; Mulla, M.M. Detection of distributed denial of service attacks in software defined networks. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 2576–2581.

122. Wang, P.; Chao, K.M.; Lin, H.C.; Lin, W.H.; Lo, C.C. An efficient flow control approach for SDN-based network threat detection and migration using support vector machine. In Proceedings of the 2016 IEEE 13th International Conference on E-Business Engineering (ICEBE), Macau, China, 4–6 November 2016; pp. 56–63.
123. Ahuja, N.; Singal, G.; Mukhopadhyay, D. DDOS attack SDN dataset. *Mendeley Data* **2020**, *1*, 17632.
124. Elsayed, M.S.; Le-Khac, N.A.; Jurcut, A.D. InSDN: A novel SDN intrusion dataset. *IEEE Access* **2020**, *8*, 165263–165284. [CrossRef]
125. CSE-CIC-IDS 2018 on AWS. Available online: https://www.unb.ca/cic/datasets/ids-2018.html (accessed on 2 March 2024).
126. IP Network Traffic Flows Labeled with 75 Apps. Available online: https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps/data (accessed on 2 March 2024).
127. Intrusion Detection Evaluation Dataset. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 2 March 2024).
128. Bakker, J.N.; Ng, B.; Seah, W.K. Can machine learning techniques be effectively used in real networks against DDoS attacks? In Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July–2 August 2018; pp. 1–6.
129. Gebremariam, A.A.; Usman, M.; Du, P.; Nakao, A.; Granelli, F. Towards e2e slicing in 5g: A spectrum slicing testbed and its extension to the packet core. In Proceedings of the 2017 IEEE Globecom Workshops (GC Wkshps), Singapore, 4–8 December 2017; pp. 1–6.
130. Hussain, M.; Shah, N.; Amin, R.; Alshamrani, S.S.; Alotaibi, A.; Raza, S.M. Software-defined networking: Categories, analysis, and future directions. *Sensors* **2022**, *22*, 5551. [CrossRef] [PubMed]
131. Ivey, J.; Yang, H.; Zhang, C.; Riley, G. Comparing a scalable SDN simulation framework built on ns-3 and DCE with existing SDN simulators and emulators. In Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, Banff Alberta, AB, Canada, 15–18 May 2016; pp. 153–164.
132. De Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using mininet for emulation and prototyping software-defined networks. In Proceedings of the 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, 4–6 June 2014; pp. 1–6.
133. Wang, S.Y.; Chou, C.L.; Yang, C.M. EstiNet openflow network simulator and emulator. *IEEE Commun. Mag.* **2013**, *51*, 110–117. [CrossRef]
134. Singla, A.; Bertino, E.; Verma, D. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 12–15 June 2019; pp. 69–74.
135. Al-Gethami, K.M.; Al-Akhras, M.T.; Alawairdhi, M. Empirical evaluation of noise influence on supervised machine learning algorithms using intrusion detection datasets. *Secur. Commun. Netw.* **2021**, *2021*, 1–28. [CrossRef]
136. Hayes, M.; Ng, B.; Pekar, A.; Seah, W.K. Scalable architecture for SDN traffic classification. *IEEE Syst. J.* **2017**, *12*, 3203–3214. [CrossRef]
137. Bianco, A.; Giaccone, P.; Kelki, S.; Campos, N.M.; Traverso, S.; Zhang, T. On-the-fly traffic classification and control with a stateful SDN approach. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
138. von Eschenbach, W.J. Transparency and the black box problem: Why we do not trust AI. *Philos. Technol.* **2021**, *34*, 1607–1622. [CrossRef]
139. Jurado-Lasso, F.F.; Marchegiani, L.; Jurado, J.F.; Abu-Mahfouz, A.M.; Fafoutis, X. A survey on machine learning software-defined wireless sensor networks (ml-SDWSNS): Current status and major challenges. *IEEE Access* **2022**, *10*, 23560–23592. [CrossRef]
140. She, C.; Yang, C.; Quek, T.Q. Cross-layer optimization for ultra-reliable and low-latency radio access networks. *IEEE Trans. Wirel. Commun.* **2017**, *17*, 127–141. [CrossRef]
141. Souri, A.; Norouzi, M.; Asghari, P.; Rahmani, A.M.; Emadi, G. A systematic literature review on formal verification of software-defined networks. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3788. [CrossRef]
142. Yao, J.; Wang, Z.; Yin, X.; Shi, X.; Wu, J.; Li, Y. Test oriented formal model of SDN applications. In Proceedings of the 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 5–7 December 2014; pp. 1–2.
143. Albert, E.; Gómez-Zamalloa, M.; Rubio, A.; Sammartino, M.; Silva, A. SDN-Actors: Modeling and verification of SDN programs. In Proceedings of the International Symposium on Formal Methods, Oxford, UK, 12 July 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 550–567.
144. Li, Y.; Yin, X.; Wang, Z.; Yao, J.; Shi, X.; Wu, J.; Zhang, H.; Wang, Q. A survey on network verification and testing with formal methods: Approaches and challenges. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 940–969. [CrossRef]
145. Aswini, K.; Reddy, U.; Nagpal, A.; Rana, A.; Abood, B.S.Z. Ensemble Learning Approaches for Big Data Classification Tasks. In Proceedings of the 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gautam Buddha Nagar, India, 1–3 December 2023; Volume 10, pp. 1545–1550.
146. Pintelas, E.; Livieris, I.E.; Pintelas, P. A grey-box ensemble model exploiting black-box accuracy and white-box intrinsic interpretability. *Algorithms* **2020**, *13*, 17. [CrossRef]
147. Amin, R.; Rojas, E.; Aqdus, A.; Ramzan, S.; Casillas-Perez, D.; Arco, J.M. A survey on machine learning techniques for routing optimization in SDN. *IEEE Access* **2021**, *9*, 104582–104611. [CrossRef]

148. Abdalzaher, M.S.; Salim, M.M.; Elsayed, H.A.; Fouda, M.M. Machine learning benchmarking for secured iot smart systems. In Proceedings of the 2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bali, Indonesia, 24–26 November 2022; pp. 50–56.
149. Schneider, S.; Satheeschandran, N.P.; Peuster, M.; Karl, H. Machine learning for dynamic resource allocation in network function virtualization. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 122–130.

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.