future internet

Article

Traffic Classification in Software-Defined Networking Using Genetic Programming Tools

Spiridoula V. Margariti , Ioannis G. Tsoulos *, Evangelia Kiousi and Eleftherios Stergiou

Department of Informatics and Telecommunications, University of Ioannina, 45110 Ioannina, Greece; smargar@uoi.gr (S.V.M.); eya.kioysh@gmail.com (E.K.); ster@uoi.gr (E.S.)
* Correspondence: itsoulos@uoi.gr

Abstract: The classification of Software-Defined Networking (SDN) traffic is an essential tool for network management, network monitoring, traffic engineering, dynamic resource allocation planning, and applying Quality of Service (QoS) policies. The programmability nature of SDN, the holistic view of the network through SDN controllers, and the capability for dynamic adjustable and reconfigurable controllersare fertile ground for the development of new techniques for traffic classification. Although there are enough research works that have studied traffic classification methods in SDN environments, they have several shortcomings and gaps that need to be further investigated. In this study, we investigated traffic classification methods in SDN using publicly available SDN traffic trace datasets. We apply a series of classifiers, such as MLP (BFGS), FC2 (RBF), FC2 (MLP), Decision Tree, SVM, and GENCLASS, and evaluate their performance in terms of accuracy, detection rate, and precision. Of the methods used, GenClass appears to be more accurate in separating the categories of the problem than the rest, and this is reflected in both precision and recall. The key element of the GenClass method is that it can generate classification rules programmatically and detect the hidden associations that exist between the problem features and the desired classes. However, Genetic Programming-based techniques require significantly higher execution time compared to other machine learning techniques. This is most evident in the feature construction method where at each generation of the genetic algorithm, a set of learning models is required to be trained to evaluate the generated artificial features.

Keywords: software-defined networking; genetic algorithms; optimization; neural networks; genetic programming

check for updates

Citation: Margariti, S.V.; Tsoulos, I.G.; Kiousi, E.; Stergiou, E. Traffic Classification in Software-Defined Networking Using Genetic Programming Tools. *Future Internet* **2024**, *16*, 338. https://doi.org/10.3390/fi16090338

Academic Editor: Sachin Sharma

Received: 28 July 2024
Revised: 12 September 2024
Accepted: 18 September 2024
Published: 19 September 2024

1. Introduction

SDN constitutes a new pathway to implement networks, to address technical issues (routing, traffic engineering, load balance, security, etc.), and provide automation and programming capabilities to the whole network. SDN promises a new era for next-generation networks that dictates a top–down approach to network design. The flexibility and ease of management of networks introduced by SDN are due to (a) the separation of the control plane from the data forwarding plane; (b) the integration of control, management, and configuration; (c) the hallmarks openness and interoperability of SDN architecture that promise vendor-neutral network implementations. A vital element of the SDN framework is the OpenFlow protocol, which is used as the standard protocol to describe the communication between the two distinct planes: the control plane and the data plane [1]. The evolution of these networks introduces the P4 (Protocol-Independent Packet Processors) as a new forwarding model, that flexibly implements the programmer's decisions about how the packets and data flows will be processed.

However, the traffic on contemporary networks is growing exponentially, and the introduction of simple rules (OpenFlow or P4) is not enough to optimally manage data traffic. One solution is to embed intelligence into physical or virtual network devices to facilitate data traffic management [2]. More specifically, it requires the implementation of methods for automated categorization of data traffic according to its features, or network traffic classification.

The term Network Traffic Classification refers to the process of identifying and associating traffic flows or data packets with the causal source (e.g., application, protocol, network services) or distinguishing them into categories based on the size of flows or priorities related to the needs of each application. Classification aims at allocating packets or flows into different and distinct categories based on an automated matching process [3,4].

Network traffic classification of data is widely used to more effectively manage computer networks, monitor their operations, estimate the Quality of Service, make network design decisions, predict future traffic patterns, provide dynamic access control, and improve service quality. It has a significant role in the efficient execution of basic network functions such as routing, traffic prediction, resource allocation, traffic engineering, load balancing, effective management functions, malicious flow identification, network traffic forecasting, and flow control [3–9]. The adoption of an appropriate classification strategy by ISPs allows them to implement efficient charging models for their customers in proportion to the resources and categories of applications they wish to use. Yet, this makes it easier for ISPs to support the Quality of Services and lawful intercept capabilities [10,11].

According to Xie et al. [6], traffic classification contributes to better network management and more efficient resource allocation in the following cases: (a) in discriminating elephant flow for implementing the best scheduling policy in data centers; (b) in matching flows with their applications (application-aware traffic classification) for optimal network management; and (c) in determining the QoS levels of traffic flows to optimize resource allocation policies.

Consequently, traffic classification is a highly important task for SDN. The accurate identification and classification of network traffic is essential to ensuring efficiency, stability, security, seamless operation, and optimal network management. Effective monitoring and management of network data traffic requires the use of efficient and highly accurate classification techniques in an effort to improve network performance. These requirements are necessitated by (a) the huge volume of web traffic; and (b) the large number of new emerging applications that claim a significant share of traffic.

Due to the high degree of scaling, traditional classification techniques fail [2]. A widely accepted solution to this problem is the consolidation of Software-Defined Networking (SDN) and machine learning. However, as pointed out in [4], the existing ML-based network traffic classifiers have to face various challenges, such as the randomization and masquerade in port numbers, the data encryption, the collection of data input, and the potential bias in the dataset. Another problem arises from the increasing data dimensionality and the large number of features. The determination of appropriate features (e.g., using K-PCA) or the extraction of features that may be used for classification are costly and hinder the traffic classification task.

In this work, we consider GenClass [12], a low-cost mechanism classification technique in terms of memory usage and evaluation time. Since this method does not involve the training of any model but only the evolution of partition rules, it can be considered a fast method. Moreover, it can use only a small part of the original features of the problem in the classification rules, speeding up the overall process. It is not dependent on the number of features, and it is capable of revealing the hidden patterns of datasets. Furthermore, GenClass has shown impressive results in solving classification problems. Thus, we propose the use of GenClass for network traffic classification in SDN environments and evaluate its performance in the case of classification traffic flows into their original application categories. We expect the results to meet the requirements for more precise and effective administration of the network.

Using a public dataset of flows from a real SDN network, we conduct experimental measurements and evaluate the ability of the method to classify traffic from six different applications. More specifically, the contribution of this work is summarized as follows:

- We consider the classification of the network traffic generated by an SDN system, and propose the development of a classifier using the GenClass tool.
- We conducted simulated experiments and evaluated the accuracy and average classification error of the proposed model.
- We provide an experimental comparison between our proposed classifier and the other well-known supervised ML methods and demonstrate the outperformance of GenClass as opposed to others. The GenClass method has managed to identify the hidden associations between the feature of the used dataset and the required classes of the problem. In addition, this method can use from the initial set of features only those that contribute significantly to the decision-making to find the desired category.
- We provide future directions to incorporate the Grammatical Evolution-based algorithms, such as GenClass, in the online process for incoming network traffic classification in real time.

The remainder of this paper is organized as follows: in Section 2 the proposed method and the used methods are fully described, in Section 3 the experimental results are listed, and finally, in Section 4, some conclusions are presented.

2. Material and Methods

2.1. Theoretical Framework

2.1.1. Operation of Software-Defined Networking

In recent years, SDN has attracted the interest of the research community and industry due to its potential for evolution, the development of new ideas, and the innovation of the network. The SDN architecture includes three distinct layers: (a) the data layer; (b) the control layer; and (c) the application implementation layer for network programming. This architecture provides many new possibilities and opportunities for future networks. A typical SDN architecture is depicted in Figure 1. The data layer includes the underlying network equipment (called switches [13]) and has as its main task the distributed forwarding of packets. The hardware independence and programmability of the data plane enable the switches to perform the desired but complex packet-switching functions. However, the control plane, a logically centralized and rich entity called controller, defines how the data must be handled. The controller usually has a wide set of modules, interacts with a collection of network applications, and has a global view of the network. The controller regulates the whole operation of the network. Thus, it makes decisions on the configuration of data traffic, defines the way of transmitting data, and allows real-time control of existing devices. The communication between the controller and application layer is achieved using the Northbound Interfaces and the communication between the controller and the network devices is realized through the Southbound APIs and the use of the OpenFlow protocol.

The fundamental requirement for achieving communication is that both the network devices and the controller must be compatible with the OpenFlow protocol [14]. OpenFlow is not used to handle network traffic.It specifies how the switches should behave and handle network traffic in accordance with controller instructions and describes the forwarding behavior of the data plane in a proactive or reactive manner [13]. Alternative P4 language can be used "to program the data plane" [15].

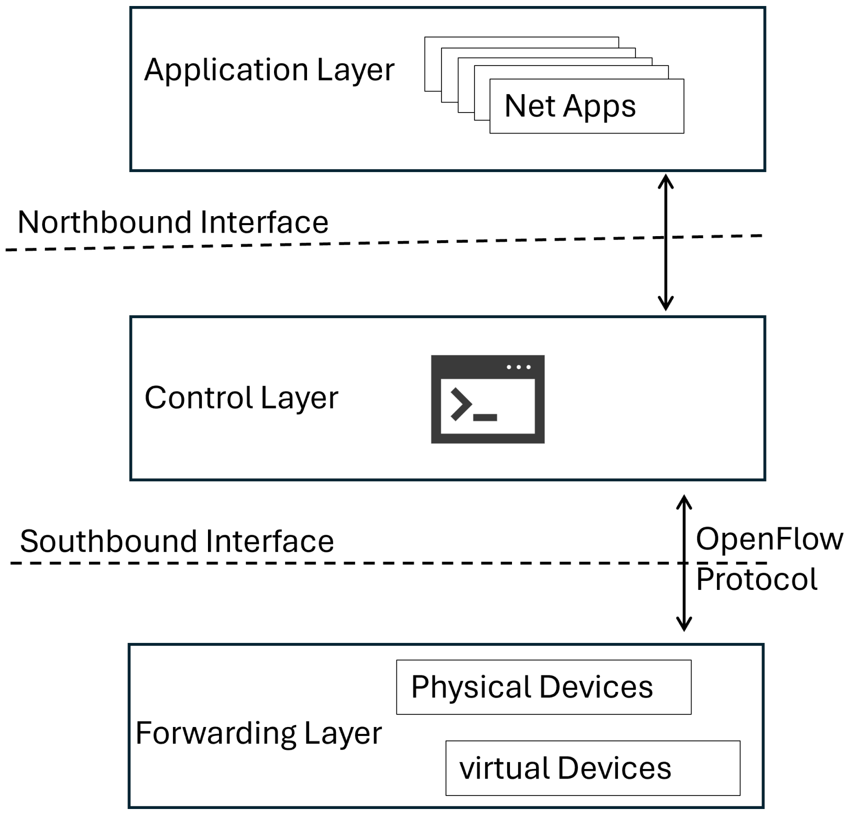

Figure 1. The typical architecture of SDN.

2.1.2. Network Traffic Classification

The wide variety of network applications introduces a large amount of data into present-day networks. Thus, traffic classification is considered essential for efficient network management, the implementation of appropriate QoS, and security mechanisms. However, effective network traffic classification is a difficult task. To distinguish packets/flows into categories, some common characteristics of packets or flows related to port, payload, correlation, behavior, and statistics are considered. All the required information is collected either from packet headers, from the packets' payload, or from the OpenFlow switches' statistics [16]. A brief description of each technique follows.

Port-based: This technique has been used successfully in the past by exploiting the port numbers provided by the Internet-Assigned Numbers Authority (IANA) for specific applications and protocols. Of course, these techniques are nowadays considered obsolete and inaccurate as most applications communicate via ephemeral (dynamic) ports or over HTTP/HTTPS (e.g., peer-to-peer apps) [17].
Deep Packet Inspection (DPI): The DPI method maps the content of packets (payload) to a signature, or specific patterns based on the application from which they originate. A DPI classifier examines the payload portion and matches it with a set of stored patterns to classify applications. It is based on four degrees of verification related to signature, syntax, protocol, and semantics. It was demonstrated that it achieves high levels of accuracy, completeness, and convergence. The main problems with the method are the inability to scale, the high cost, the increased complexity, the inability to apply to encrypted traffic, and emerging privacy issues. There is also a requirement to continuously update stored signatures or templates to include new application templates [18].
Statistical-based classification: Unlike other approaches, statistical-based classification relies on statistical features extracted from packets or traffic flows of network data in combination with machine learning methods. The extracted features, such as packet duration, packet length, time between consecutive arrivals, flow duration time, and flow idle time, are considered unique for each traffic type and are used as inputs to the ML models that decide

the classification. This classification approach achieves high accuracy but requires a lot of time to process the data. In general, the performance of these techniques depends largely on the quality of the collected features [10].
Behavioral classification: In this approach, the point of observation is the end-host, and using appropriate tools, the entire traffic received by it is examined to detect the type of application or protocol running on the target endpoint [19]. This method is also based on the collection of statistics from traffic. The information, such as the distribution of packet sizes, the time between successive packet arrivals, etc., is used to identify the application generating the data. This technique does not need to examine the payload or the encrypted data, but it requires a large number of samples to achieve high accuracy.

2.1.3. ML-Based Traffic Classification Methods in SDN and Related Works

A widely accepted solution for SDN traffic classification is the use of machine learning (ML) techniques, which exploit extracted statistical data and distinguish data flows into several categories. For the categorization of flows, mainly their statistical data are utilized and, based on these data, the different flows are distinguished into several classes or groups.

The development of ML algorithms for SDN network traffic classification is implemented (a) at the control layer using a classifier module and (b) at the data layer using P4 language [20]. In the first case, the SDN controller collects information and, in cooperation with the application layer, exploits it to implement forwarding or access policies. The process is completed in three steps [10]: (1) the data are sent to the controller by OpenFlow Protocol; (2) the controller extracts the desired features, which feed the classifier; (3) the classifier performs the task of classification. In the second case, the classification is conducted at the data level, and the P4 programming language is used to add ML mechanisms to the switches to categorize the flows, making it more effective and allowing it to identify flows as soon as possible.

The majority of research approaches that deal with network traffic classification challenges in SDN leverage data collected by the OpenFlow protocol and extract the appropriate features to feed the input of classification process. These features include (a) observables such as packet size, interval arrival time, source and destination IP/MAC/Port, flow duration, and byte count; (b) defined and calculated features such as mean and variance of traffic characteristics; (c) scalars, such as minimum and maximum values. Then, on selected or on the set of features supervised, semi-supervised, or unsupervised, reinforcement learning [6] classification methods are applied to classify the data traffic flows in terms of (a) traffic protocols (e.g., http, smtp, ssh, p2p, dns); (b) application type (e.g., DDoS, FTP, and video streaming, Youtube, video, Facebook, Linkedin, Skype); (c) flow size (e.g., elephants or mice); (d) QoS classes (e.g., interactive apps and bulky data transfer).

The accuracy and quality of classification results depend on the training datasets. However, obtaining standardized and high-quality data that capture the actual traffic of a network is very difficult [6]. Researchers exploit publicly available datasets, generate datasets of the desired traffic through simulations, or collect data from real networks themselves.

The traffic classification in SDN has been extensively studied [10,21]. Among the most important issues explored and solutions proposed, we distinguish the following main categories: application identification, security issues, QoS categories, and network management, including traffic management and resource management. For example, a plethora of researchers have tried to classify SDN network traffic according to the type of application aimed at improving network efficacy and efficiency.

Ashour et al. [22] used a real-time dataset to assess how efficient ML models (Random Forest (RF), K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), and Naive Bayes (NB)) perform traffic classification in six common internet protocols and services. The DT algorithm is experimentally proven to be the most efficient model, outperforming the others with an accuracy of 99.8%.

Nunez-Agurto et al. [23] also applied deep learning methods to classify network traffic into five categories (file transfer, instant messaging, multimedia, VoIP, and attacks). Using two publicly available datasets, namely InSDN and VPN-nonVPN, they applied four RNN algorithms to classify traffic, seeking to enhance QoS and security in SDN. Although the proposed model is distinguished for its efficiency, it suffers from training difficulties and high computational costs. A similar objective was also pursued by Raikar et al. [2], namely, classifying traffic according to its type (HTTP, mail, and streaming). Using three common machine learning algorithms, SVM, NB, and the nearest centroid, they achieved an accuracy of over 90%.

The advanced characteristics of SDN, namely the overall network view, programmability, and functionality splitting of the data and control plane, have piqued the interest of the research community in implementing new QoS provisioning concepts. Pradhan et al. [24] proposed a hybrid classifier implemented by a neural network combined with a particle swarm optimization algorithm (NN-PSO) to improve QoS in an underwater sensor network. Using a publicly available dataset and implementing three different classifiers (Feed-Forward Neural Network (FNN), LR, and NB), they achieve enhanced accuracy.

Furthermore, SDN architecture promises ease and flexibility in network management, and researchers are applying traffic taxonomy to contribute to this direction. Vulpe and Dobrin [25] developed and implemented a traffic classification model to classify data traffic for management purposes. The model can be run in real time, is implemented on the controller side, and uses six classical classification algorithms to categorize packets.

The drawbacks of this approach include the limitations of non-typical traffic detection, the number of features used, and the role of data normalization in the performance of the classification model. Recognizing the problems of port-based and DPI techniques, Geremew and Ding [26] have proposed a deep learning-based solution for the dynamic discovery of elephant flows. This classification enables optimal and efficient resource allocation and ensures optimal QoS in the SDN system. The flows are distinguished into two categories by applying DNN, CNN, LSTM, and deep autoencoder algorithms. For the experimental evaluation, the measurable characteristics (such as flow size and packet size) were collected from three distinct datasets. The range of the flow detection rate is 98.17% to 98.78%. The method is computationally demanding, but it is quite resilient and can be implemented on the controller side. In [9], the authors attempt to optimize traffic engineering by adding a traffic classification module to the RYU controller.

Despite the benefits of SDN technology and the ability to manage from a single central point, it is still vulnerable to attacks, tampering, and disasters. According to Spyrou et al. [27], implementing a traffic classification technique can help detect DDoS attacks on an SDN network. Thus, they propose the use of GenClass, a tool based on classification rules generated by the Grammatical Evolution method. The authors demonstrate experimentally that the results of GenClass are better than other similar classification methods. Table 1 summarizes the related work. For every study, we state information about the network problem examined by the authors, the ML model and ML algorithm that were used, the type of traffic that they tried to classify, the used dataset, the feauters that were chosen as input to the classifier, the target performance metrics, the used SDN controller (if any), the network topology (if any), and the metric accuracy. The metric accuracy represents the best result of all the investigated ML algorithms.

As shown in Table 1, there are a large number of efforts in the scientific literature to study traffic classification in the SDN emerging paradigm. Most of them utilize supervised ML classical algorithms such as SVM, k-MN, DT, LR, etc., while others adopt more recent approaches such as GenClass. Only a few of the researchers implement measurements using SDN controllers and particular topologies. The most commonly used metrics are accuracy, precision, recall, and F1-score. It is worth noting that accuracy can be improved, and this is achieved in our work by using the GenClass tool.

Table 1. A summary of research works and their findings.

Authors and Year	Challenge	ML Model	ML Algorithms	Type of Traffic	Dataset	Features	Metrics	Controller	Topology	Accuracy
Ganesan et al. [28], 2021	QoS issues	Supervised learning	RF, KNN, NN-MLP, NB, LR, SVM	Voice, video, IoT (IoT traffic types), HTTP	UNSW dataset (publicly available)	Statistical features (port numbers, IP src, IP dst, DNS, NTP, packet size, etc.)	TP, FP, FN, TN, accuracy, precision, recall, F1-score		Tree and branch topology	99%
Wassie et al. [26], 2023	Network management	Deep Learning	DNN, CNN, LSTM, and deep autoencoder	Real-time apps	NIMS, SDN datasets	Low size, total packet size, protocol, app type, flow duration, and more (23 features)	Accuracy, loss metrics, run time	RYU		99.12%
Lin et al. [9], 2023	Network management	Supervised learning	RF	Chat, email, file transfer, streaming, P2P, VoIP, HTTP	ISCX-VPN-NonVPN-2016	Average packet inter-arrival time, distribution of packet sizes or extracted features	F-score, throughput, confusion matrix	RYU	Simple diamond	
Ashour et al. [22], 2024	Application identification	Supervised learning	KNN, LR, RF, DT, NB, and SVM	WWW, DNS, FTP, ICMP, P2P, and VOIP	SDN-traffic, Kaggle	Quantity of packets, avg. transmission time, number of instantly transmitted packets	Accuracy, F1-score, recall, precision, and training time		Straight-forward topology	99.8%
Nunez-Agurto, et al. [23], 2024	Application identification and security issues	Deep Learning	LSTM, BiLSTM, GRU, and BiGRU	Multimedia, VoIP, Instant message, File transfer, Attack	InSDN and ISCX-VPNNoVPN	12 features	Accuracy, precision, recall, and F1-Score			99.65%
Perera et al. [21], 2020	Network management	Unsupervised and supervised ML	RF, KNN, DT, SVM	Four types of traffic	Publicly available	MAC addresses and ports, flow duration, flow byte count, flow packet count, and average packet size	Accuracy	RYU	Simple	96.37%
Pradhan et al. [24], 2022	QoS issues	Semi-supervised machine learning	Feed-forward NN, NB, and LR	HTTP, video streaming, FTP, P2P, instant messaging	ANT and Kaggle dataset	Packet size, packet inter-arrival time	Confusion matrix, precision, recall, and F-Measure			
Raikar et al. [2], 2020	Application identification	Supervised learning	SVM, NB, nearest centroid	HTTP, mail, streaming	Own collected data	srcip, srcport, dstip, dstport, proto, total fpackets, total fvolume, total bpackets	Accuracy, precision, recall, F-score, training error, training and testing time	POX	Simple, linear, tree	96.79%

Table 1. *Cont.*

Authors and Year	Challenge	ML Model	ML Algorithms	Type of Traffic	Dataset	Features	Metrics	Controller	Topology	Accuracy
Spyrou et al. [27], 2023	Security issues	Supervised learning	NB, KNNs, RF as opposed to GenClass	Dataset of normal and malicious traffic	DDoS SDN dataset	23 features extracted from switches	Average class error			
Vulpe et al. [25], 2023	Network management (traffic monitoring)	Supervised learning	LR, KNN, NB, SVM, DT, ANN	Ping, DNS, Telnet, and voice (real time)	Own collected data	Delta packets, delta bytes, packet statistics, instant bytes per second, average bytes per second, time	Accuracy, precision, recall, F1-score	RYU	mininet topologies	97.94%
Our work	Application identification	Supervised learning	MLP (BFGS), FC2 (RBF), FC2 (MLP), GENCLASS	DNS, WWW, FTP, P2P, ICMP, VOIP	SDN-dataset	Flow features	Average classification error			99.42%

2.2. Proposed Approach

The current section describes the used dataset as well as the used machine learning methods. The methods used here were also utilized in the paper of Mpouziotas et al. [29].

2.2.1. The Dataset

The used dataset, that is 'SDN-traffic', was obtained from the Kaggle platform. This dataset was created by gathering data of an SDN architecture and made available by [22]. It includes traffic data for an SDN network. It consists of 4234 records and 65 features stored in a CSV file describing the data traffic of six different network applications, namely DNS, WWW, FTP, P2P, ICMP, and VoIP. The features included the flow ID, the source and destination IP address, the source and destination port, the transport layer protocol, and other traffic characteristics collected by OpenFlow counters. The detailed list of the features has been visualized in Table 2. This particular dataset was selected due to its real nature, diversity, and richness, which enables it to effectively identify various traffic behaviors.

Table 2. A subset of features included in the SDN-traffic dataset.

Features	Description	Features
forward_pl	Package size in bytes	reverse_pl
forward_piat	Packet arrival interval in seconds	reverse_piat
forward_pps	Packets per second	reverse_pps
forward_bps	Bytes per second	reverse_bps
forward_pl_mean	Average packet size in bytes	reverse_pl_mean
forward_piat_mean	Average packet arrival interval in seconds	reverse_piat_mean
forward_pps_mean	Average number of packets per second	reverse_pps_mean
forward_bps_mean	Average number of bytes per second	reverse_bps_mean
forward_pl_var	Variance of packet size in bytes	reverse_pl_var
forward_piat_var	Variance of packet arrival interval in seconds	reverse_piat_var
forward_pps_var	Variance of the number of packets per second	reverse_pps_var
forward_bps_var	Variance of the number of bytes per second	reverse_bps_var
forward_pl_q1	1st quartile of packet size in bytes	reverse_pl_q1
forward_pl_q3	3rd quartile of packet size in bytes	reverse_pl_q3
forward_piat_q1	1st quartile of the packet arrival interval in seconds	reverse_piat_q1
forward_piat_q3	3rd quartile of the packet arrival interval in seconds	reverse_piat_q3
forward_pl_max	Maximum packet size in bytes	reverse_pl_max
forward_pl_min	Minimum packet size in bytes	reverse_pl_min
forward_piat_max	Maximum packet arrival interval in seconds	reverse_piat_max
forward_piat_min	Minimum packet arrival interval in seconds	reverse_piat_min
forward_pps_max	Maximum number of packets per second	reverse_pps_max
forward_pps_min	Maximum number of packets per second	reverse_pps_min
forward_bps_max	Maximum number of bytes per second	reverse_bps_max
forward_bps_min	Minimum number of bytes per second	reverse_bps_min
forward_duration	Duration	reverse_duration
forward_size_packets	Size of packets	reverse_size_packets
forward_size_bytes	Size of bytes	reverse_size_bytes
	Application type	Category

The distribution of used applications included in the dataset has been listed in Table 3. From Table 3, it became apparent that WWW is carrying the majority of data traffic, while FTP and DNS protocols convey the lowest portion of traffic. By classifying network traffic into categories per application type, we aim to enhance network performance. The SDN controller, being aware of the running applications (or protocols) and their requirements, can make informed decisions about network, resource, and data traffic management.

Table 3. The distribution of the various application types.

Application	Number of Flows	Proportion
WWW	2441	57.65%
P2P	710	16.77%
ICMP	409	9.66%
VOIP	256	6.05%
FTP	217	5.53%
DNS	184	4.35%

2.2.2. Machine Learning Methods

A series of machine learning methods were used as classifiers for the provided dataset:

1. GENCLASS: GenClass is a promising classification tool based on genetic programming. It utilizes the Grammatical Evolution method [30] to generate classification rules for a Backus–Naur Form. Its experimental evaluation on a number of different datasets demonstrates its superiority in the majority of cases. This technique was initially presented in the work of Tsoulos [12]. Furthermore, a software that implements this method was also published recently [31]. A short description of this method can be found in Section 2.3.
2. MLP: A combination of a standard neural network [32,33] and Broyden–Fletcher–Goldfarb–Shanno (BFGS), i.e., an iterative optimization algorithm [34]. BFGS is used to optimize the parameters of the MLP model and is characterized by generality and ease of implementation.
3. SVM: a Support Vector Machine method [35,36] was used as a machine learning model in the conducted experiments.
4. TREE: the well-known method of Decision Trees [37,38] was also used as a classification method.
5. FC2 (RBF): Represents the method of feature construction with the assistance of Grammatical Evolution, initially presented in the work of Gavrilis et al. [39]. In this approach, $N_F = 2$ artificial features were created using Grammatical Evolution and subsequently evaluated using the Radial Basis Function (RBF) network [40,41]. The creation of new artificial features from the existing ones on the one hand aims to reduce the dimension of the problem and help to optimize the generalization ability of machine learning models and on the other hand to find hidden correlations that may exist between the existing features which would lead to more efficient training machine learning models.
6. FC2 (MLP): Represents the application of feature construction technique to create two artificial features for the provided dataset. These features were evaluated using a neural network. A brief description of the feature construction method can be found in Section 2.4.

In this work, two techniques based on Grammatical Evolution were selected for comparative study: the construction of artificial features and the construction of classification rules. The first technique aims to reduce, in principle, the number of features required to successfully classify the data but also to discover hidden associations between existing features that could not be discovered otherwise. The second technique, which from the experimental results appeared to be more reliable, creates classification rules from the existing characteristics on the one hand and on the other hand can be used as a selector of those characteristics that contribute more to the correct classification of the patterns.

2.3. The GenClass Method

The main steps of the used method are provided below.

1. **Initialization Step:**
   (a) **Set** the parameters of the method: N_c for number of chromosomes, N_g for maximum number of allowed generations, p_s for the selection rate, and $p_m \leq 1$ for the mutation rate.
   (b) **Obtain** the train set $\text{TR} = \{x_i, y_i\},\ i = 1 \ldots\ M$ of the used dataset.
   (c) **Set** $k = 0$, the iteration number.
2. **Fitness Calculation Step:**
   (a) **For** $i = 1 \ldots N_c$ **do**
      i. **Create** a classification program C_i using the Grammatical Evolution procedure and the grammar defined in [12].
      ii. **Calculate** the fitness value f_i of chromosome i as

$$f_i = \sum_{j=1}^{M} \left(C_i(x_j) \neq y_j\right) \tag{1}$$

   (b) **EndFor**.
3. **Genetic Operations Step:**
   (a) **Selection procedure.** All chromosomes are sorted according to their fitness values calculated before and the first $(1 - p_s) \times N_C$ of chromosomes with the lowest fitness values are transferred intact to the next generation. The remaining chromosomes are substituted by offsprings produced during the crossover procedure.
   (b) **Crossover procedure**. In order to produce new offsprings, the tournament selection is used to select a pair of (z, w) chromosomes from the current population. This set of selected chromosomes will produce two new offsprings $\tilde{z}$ and $\tilde{w}$ using one-point crossover.
   (c) **Perform** the mutation procedure. During this procedure, a random number $r, r \in [0, 1]$ is drawn for each element of every chromosome. If $r \leq p_m$, then the corresponding element is altered randomly.
4. **Termination Check Step:**
   (a) **Set** $k = k + 1$;
   (b) **If** $k \geq N_g$, terminate; or else, go to the Fitness Calculation Step.

2.4. The Feature Construction Method

In this procedure, a series of artificial features can be created from the original ones using Grammatical Evolution. The main steps of this method are as follows:

1. **Initialization Step:**
   (a) **Set** the parameters of the method: N_c for number of chromosomes, N_g for maximum number of allowed generations, p_s for the selection rate, $p_m \leq 1$ for the mutation rate, and N_f the number of desired features that should be constructed.
   (b) **Obtain** the train set $\text{TR} = \{x_i, y_i\},\ i = 1 \ldots M$ of the used dataset.
   (c) **Set** k = 0 as the generation number.
2. **Fitness Calculation Step**:
   (a) **For** $i = 1 \ldots N_c$ **do**
      i. **Create** N_f artificial features from the original ones using Grammatical Evolution.
      ii. Create the dataset TR_i from the original one, TR, using the new features.
      iii. Train an RBF network $r_i(x, w)$ on the TR_i dataset. This network is preferred over other neural networks because of its very fast training method.

iv. **Calculate** the corresponding fitness value f_i as

$$f_i = \sum_{j=1}^{M} \left(r_i(x_j, w) - y_j\right)^2 \tag{2}$$

(b) Select **EndFor**

3. **Genetic Operations Step:** use the same genetic operators as in GenClass method provided in Section 2.3.
4. **Termination Check Step:**
   (a) **Set** $k = k + 1$;
   (b) **If** $iter \geq N_g$, terminate; or else return to the Fitness Calculation Step.

3. Results

The used software was coded in ANSI-C++ with the assistance of the freely available GenClass version 1.0 software to create classification programs. The software is available from the relevant url https://github.com/itsoulos/GenClass/ (accessed on 23 July 2024). Furthermore, the software QFc version 1.0 is used to create artificial features with the assistance of Grammatical Evolution. The software [42] is available freely https://github.com/itsoulos/QFc/ (accessed on 23 July 2024). All experiments were conducted on an AMD Ryzen 5950X with 128 GB of RAM, running Debian Linux. To validate the results, ten-fold cross-validation was used. In this technique, the original dataset was divided into ten partitions. Nine of the partitions created were used for training and the remainder for testing, and this process was repeated ten times. The final classification error was the average test error for the test partitions. All the experiments were executed 30 times, using different seed for the random generator each time and the average classification error is reported. The simulation parameters for the used optimization techniques are listed in Table 4. The values of these parameters have been used in the past in a number of research publications and represent a compromise between the reliability of the techniques and the speed of their execution.

Table 4. The values used for the parameters in the conducted experiments.

Parameter	Meaning	Value
N_c	Number of chromosomes/particles	500
N_g	Maximum number of allowed generations	200
N_f	Number of produced artificial features	2
H	Number of weights for neural network	10
p_s	Selection rate	0.90
p_m	Mutation rate	0.05

The experimental results are outlined in Table 5. The rows in this table have the following meaning:

1. The row MLP stands for a neural network [32,33] with H hidden nodes that was trained with the application of the BFGS optimization method [34].
2. The row SVM stands for the application of Support Vector Machine on the proposed dataset. The LibSVM implementation [43] was used in the conducted experiments.
3. The row TREE refers to the application of Decision Trees to the objective problem using the freely available software https://github.com/ikeofilic1/dtrees (accessed on 28 August 2024).
4. The row FC2 (RBF) denotes the application of the feature construction technique. The method was used to construct N_F artificial features and these features were evaluated using an RBF network with H weights.

5. The row FC2 (MLP) represents the application of the feature-construction technique to create N_F artificial features. The produced features are evaluated using a neural network with H hidden nodes.
6. The row GENCLASS denotes the usage of the GenClass method to produce classification rules.

Table 5. Experimental results using a variety of machine learning methods.

Method	Classification Error
MLP	16.33%
SVM	20.38%
TREE	7.64%
FC2(RBF)	13.67%
FC2(MLP)	9.35%
GENCLASS	0.58%

As was observed from the results described in Table 5, the GenClass method results in a very low average classification error, which is 0.58%, in contrast to other applied methods where the error is much higher and ranges from 7.64% to 20.38%. As mentioned above, the training of the model was conducted through cross-validation, which, according to Belkadi et al. [44], leads to a decrease in the correct classification rate. However, this assumption for the GenClass case is not confirmed.

Furthermore, precision and recall were recorded for all machine learning methods and the results are depicted in Table 6. Furthermore, in this case, the superiority of the GenClass technique is obvious over the rest of the machine learning methods.

Table 6. Precision and recall for the used methods. The metric precision is the proportion of all classifications that were correct and the metric recall represents the proportion of all actual positive patterns that were classified correctly as positives.

Method	Precision	Recall
MLP	0.37	0.68
SVM	0.51	0.53
TREE	0.51	0.93
FC2(RBF)	0.62	0.69
FC2(MLP)	0.85	0.92
GENCLASS	0.96	0.95

The results are very encouraging, and thanks to the innovation of SDN, the GenClass tool could be integrated into the control plane of an SDN system. This decision was made because at the control plane (a) it becomes easier to collect statistical information about the network traffic and (b) it facilitates the integration of GenClass into the controller which adapts the packet routing decisions improving the network performance. However, the time of techniques based on Grammatical Evolution is significantly longer compared to the rest of the techniques, as shown in the graph of Figure 2. Feature construction using Grammatical Evolution is the method that requires the most significant execution time, since at each generation of the genetic algorithm, a series of neural networks are trained to evaluate the generated artificial features. For every generation of the feature construction technique, a series of neural networks should be trained on the modified training set created using the candidate artificial features. This step is necessary in order to evaluate the candidate features and the fitness of each chromosome will be the training error of the neural network trained on the modified training set. On the other hand, the GenClass method appears to have a significantly increased execution time than the training of an artificial neural network, but since it does not include any parametric model training, the execution time required is significantly shorter than in the case of feature construction.The GenClass method simply creates classification rules that do not need any training and the fitness of the corresponding chromosome will be the classification error of the classification rule created by Grammatical Evolution.

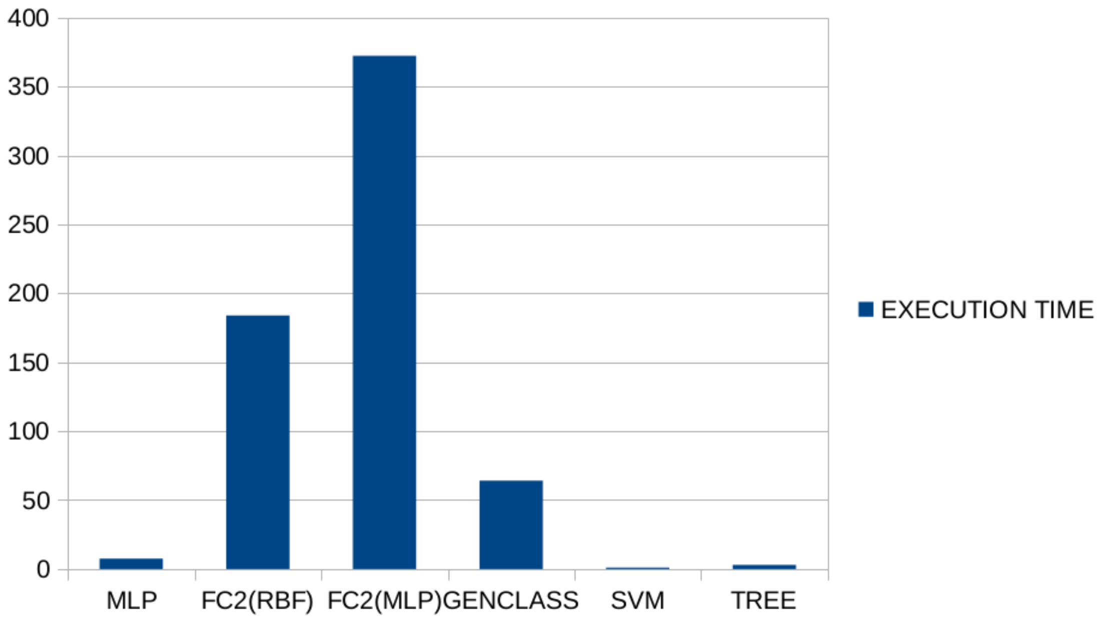

Figure 2. A schematic representation of the execution time for each machine learning method. The vertical axis represents average execution time for each method measured in seconds.

According to Anastaspoulos et al. [31], the runtime can be reduced by using parallel processing techniques, such as MPI [45] or OPENMP [46]. Therefore, multi-threaded controllers such as Floodlight, Beacon, Maestro, OpenMul, ODL, and ONOS [47] are potential candidates for the integration of ML classification models.

4. Conclusions

This work focuses on traffic classification in an SDN environment. Traffic classification is an essential tool for network management, Quality of Service (QoS), security, network optimization, and cost reduction. Thus, we strive to find an efficient solution for SDN traffic classification. The separation of the control plane and the data plane in SDN facilitates the integration of classification tools either in the control plane or the data plane. We propose the use of the GenClass tool, for which, using the SDN-traffic.csv dataset, we find very high accuracy in traffic data classification in the application classification.

In reality, networks are more complex, and data traffic patterns are influenced by user sharing, scale, the advent of new protocols, application intelligence, and network stability [10]. The performance of the proposed classifier creates new opportunities for handling such situations and motivates further investigation. For example, for future work, we will explore the integration of GenClass into a controller and its real-time operation. Moreover, we will investigate the applicability of this classifier in the infrastructure layer of SDN using the P4 language.

As it is clear from past studies and as pointed out by several researchers, a key problem is the collection of the data to be used as inputs to the traffic classification models. The collection of data requires systematic monitoring of the network. Such a process can be costly, introduce overhead, and also determine the quality and accuracy of the results. An emerging approach for network monitoring [48,49] that is applicable to SDN networks and provides solutions to the above problems is network tomography [50]. Given the increasing scale and complexity of modern networks, the use of network tomography to obtain flow characteristics presents significant advantages such as low overhead, directness, and improvement of network traffic. Thus, future work could be a thorough investigation of the coexistence of network tomography and network traffic classification in SDN systems.

Moreover, we will investigate the applicability of this classifier in the infrastructure layer of SDN using the P4 language.

Author Contributions: I.G.T. and E.K. conceived the idea and methodology and supervised the technical part regarding the software. I.G.T. conducted the experiments, employing several datasets, and provided the comparative experiments. E.S. and S.V.M. performed the statistical analysis and all other authors prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been financed by the European Union: Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan under the call RESEARCH—CREATE—INNOVATE, project name "iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

References

1. Göransson, P.; Black, C.; Culver, T. Software defined networks. In *A Comprehensive Approach*; Elsevier: Amsterdam, The Netherlands, 2017; p. 409.
2. 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]
3. Foremski, P. On different ways to classify Internet traffic: A short review of selected publications. *Theor. Appl. Inform.* **2013**, *25*, 119–136.
4. Azab, A.; Khasawneh, M.; Alrabaee, S.; Choo, K.K.R.; Sarsour, M. Network traffic classification: Techniques, datasets, and challenges. *Digit. Commun. Netw.* **2024**, *10*, 676–692. [CrossRef]
5. Xu, C.; Qin, D.; Song, F. A survey of SDN traffic management research. In Proceedings of the 2022 11th International Conference on Communications, Circuits and Systems (ICCCAS), Singapore, 13–15 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 231–236.
6. 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. Tutorials* **2018**, *21*, 393–430. [CrossRef]
7. Amanowicz, M.; Jankowski, D. Detection and classification of malicious flows in software-defined networks using data mining techniques. *Sensors* **2021**, *21*, 2972. [CrossRef]
8. 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. [CrossRef]
9. Lin, C.Y.; Huang, H.Y. A Traffic Classification Based Traffic Engineering Framework in Software-Defined Networking. *SSRN* **2023**, 4394498.
10. 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; IEEE: Piscataway, NJ, USA, 2018; pp. 200–206.
11. Shafiq, M.; Yu, X.; Laghari, A.A.; Yao, L.; Karn, N.K.; Abdessamia, F. Network traffic classification techniques and comparative analysis using machine learning algorithms. In Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 14–17 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2451–2455.
12. Tsoulos, I.G. Creating classification rules using grammatical evolution. *Int. J. Comput. Intell. Stud.* **2020**, *9*, 161–171.
13. Nunes, B.A.A.; Mendonca, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 1617–1634. [CrossRef]
14. 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]
15. Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–95. [CrossRef]
16. Ajaeiya, G.A.; Adalian, N.; Elhajj, I.H.; Kayssi, A.; Chehab, A. Flow-based Intrusion Detection System for SDN. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017; pp. 787–793. [CrossRef]
17. Qazi, Z.A.; Lee, J.; Jin, T.; Bellala, G.; Arndt, M.; Noubir, G. Application-awareness in SDN. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 487–488.

18. 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; IEEE: Piscataway, NJ, USA,, 2016; pp. 1–5.
19. Valenti, S.; Rossi, D.; Dainotti, A.; Pescapè, A.; Finamore, A.; Mellia, M. Reviewing traffic classification. In *Data Traffic Monitoring and Analysis: From Measurement, Classification, and Anomaly Detection to Quality of Experience*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 123–147.
20. Kamath, R.; Sivalingam, K.M. Machine Learning based Flow Classification in DCNs using P4 Switches. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; pp. 1–10. [CrossRef]
21. 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; Revised Selected Papers 2; Springer: Berlin/Heidelberg, Germany, 2020; pp. 28–39.
22. Ashour, M.M.; Yakout, M.A.; AbdElhalim, E. Traffic Classification in Software Defined Networks based on Machine Learning Algorithms. *Int. J. Telecommun.* **2024**, *4*, 1–19.
23. Nuñez-Agurto, D.; Fuertes, W.; Marrone, L.; Benavides-Astudillo, E.; Coronel-Guerrero, C.; Perez, F. A Novel Traffic Classification Approach by Employing Deep Learning on Software-Defined Networking. *Future Internet* **2024**, *16*, 153. [CrossRef]
24. Pradhan, B.; Srivastava, G.; Roy, D.S.; Reddy, K.H.K.; Lin, J.C.W. Traffic classification in underwater networks using sdn and data-driven hybrid metaheuristics. *ACM Trans. Sens. Netw. (TOSN)* **2022**, *18*, 1–15. [CrossRef]
25. Vulpe, A.; Dobrin, C.; Stefan, A.; Caranica, A. AI/ML-based real-time classification of Software Defined Networking traffic. In Proceedings of the 18th International Conference on Availability, Reliability and Security, Benevento, Italy, 29 August–1 September 2023; pp. 1–7.
26. Wassie Geremew, G.; Ding, J. Elephant Flows Detection Using Deep Neural Network, Convolutional Neural Network, Long Short-Term Memory, and Autoencoder. *J. Comput. Netw. Commun.* **2023**, *2023*, 1495642. [CrossRef]
27. Spyrou, E.D.; Tsoulos, I.; Stylios, C. Distributed Denial of Service Classification for Software-Defined Networking Using Grammatical Evolution. *Future Internet* **2023**, *15*, 401. [CrossRef]
28. Ganesan, E.; Hwang, I.S.; Liem, A.T.; Ab-Rahman, M.S. SDN-enabled FiWi-IoT smart environment network traffic classification using supervised ML models. *Photonics* **2021**, *8*, 201. [CrossRef]
29. Mpouziotas, D.; Besharat, J.; Tsoulos, I.G.; Stylios, C. AliAmvra—Enhancing Customer Experience through the Application of Machine Learning Techniques for Survey Data Assessment and Analysis. *Information* **2024**, *15*, 83. [CrossRef]
30. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
31. Anastasopoulos, N.; Tsoulos, I.G.; Tzallas, A. GenClass: A parallel tool for data classification based on Grammatical Evolution. *SoftwareX* **2021**, *16*, 100830. [CrossRef]
32. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
33. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
34. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547–566. [CrossRef]
35. Deris, A.M.; Zain, A.M.; Sallehuddin, R. Overview of support vector machine in modeling machining performances. *Procedia Eng.* **2011**, *24*, 308–312. [CrossRef]
36. Zhang, D. Support vector machine. In *Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval*; Springer International Publishing: Cham, Switzerland, 2021; pp. 201–228.
37. Kotsiantis, S.B. Decision trees: A recent overview. *Artif. Intell. Rev.* **2013**, *39*, 261–283. [CrossRef]
38. Charbuty, B.; Abdulazeez, A. Classification based on decision tree algorithm for machine learning. *J. Appl. Sci. Technol. Trends* **2021**, *2*, 20–28. [CrossRef]
39. Gavrilis, D.; Tsoulos, I.G.; Dermatas, E. Selecting and constructing features using grammatical evolution. *Pattern Recognit. Lett.* **2008**, *29*, 1358–1365. [CrossRef]
40. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef]
41. Yu, H.; Xie, T.; Paszczynski, S.; Wilamowski, B.M. Advantages of Radial Basis Function Networks for Dynamic System Design. *IEEE Trans. Ind. Electron.* **2011**, *58*, 5438–5450. [CrossRef]
42. Tsoulos, I.G. QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution. *Algorithms* **2022**, *15*, 295. [CrossRef]
43. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2011**, *2*, 1–27. [CrossRef]
44. Belkadi, O.; Vulpe, A.; Laaziz, Y.; Halunga, S. ML-Based Traffic Classification in an SDN-Enabled Cloud Environment. *Electronics* **2023**, *12*, 269. [CrossRef]
45. Gropp, W.; Lusk, E.; Doss, N.; Skjellum, A. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Comput.* **1996**, *22*, 789–828. [CrossRef]
46. Chandra, R. *Parallel Programming in OpenMP*; Morgan Kaufmann: Cambridge, MA, USA, 2001.

47. Zhu, L.; Karim, M.M.; Sharif, K.; Xu, C.; Li, F.; Du, X.; Guizani, M. SDN controllers: A comprehensive analysis and performance evaluation study. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–40. [CrossRef]
48. Zhang, P.; Zhao, Y.; Wang, Y.; Jin, Y. Enhancing network performance tomography in software-defined cloud network. *IEEE Commun. Lett.* **2016**, *27*, 832–835. [CrossRef]
49. Craig, A.; Nandy, B.; Lambadaris, I.; Ashwood-Smith, P. Load balancing for multicast traffic in SDN using real-time link cost modification. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 5789–5795.
50. Kakkavas, G.; Stamou, A.; Karyotis, V.; Papavassiliou, S. Network tomography for efficient monitoring in SDN-enabled 5G networks and beyond: Challenges and opportunities. *IEEE Commun. Mag.* **2021**, *59*, 70–76. [CrossRef]