symmetry

MDPI

Article

Effects of Machine Learning Approach in Flow-Based Anomaly Detection on Software-Defined Networking †

Samrat Kumar Dey [1,*] and Md. Mahbubur Rahman [2]

[1] Department of Computer Science and Engineering, Dhaka International University (DIU), Dhaka 1205, Bangladesh

[2] Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST), Dhaka 1216, Bangladesh; mahbubucse@yahoo.com

* Correspondence: sopnil.samrat@gmail.com; Tel.: +88-01823-26-79937

† This paper is an extended version of our paper published in 2018 IEEE 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 416–421.

Received: 9 November 2019; Accepted: 10 December 2019; Published: 18 December 2019

Abstract: Recent advancements in software-defined networking (SDN) make it possible to overcome the management challenges of traditional networks by logically centralizing the control plane and decoupling it from the forwarding plane. Through a symmetric and centralized controller, SDN can prevent security breaches, but it can also bring in new threats and vulnerabilities. The central controller can be a single point of failure. Hence, flow-based anomaly detection system in OpenFlow Controller can secure SDN to a great extent. In this research, we investigated two different approaches of flow-based intrusion detection system in OpenFlow Controller. The first of which is based on machine-learning algorithm where NSL-KDD dataset with feature selection ensures the accuracy of 82% with random forest classifier using the gain ratio feature selection evaluator. In the later phase, the second approach is combined with a deep neural network (DNN)-based intrusion detection system based on gated recurrent unit-long short-term memory (GRU-LSTM) where we used a suitable ANOVA F-Test and recursive feature elimination selection method to boost classifier output and achieve an accuracy of 88%. Substantial experiments with comparative analysis clearly show that, deep learning would be a better choice for intrusion detection in OpenFlow Controller.

Keywords: software-defined networking; random forest; gain ratio; GRU-LSTM; ANOVA F-test; OpenFlow Controller; machine learning

1. Introduction

Current Internet protocol (IP) networks are increasingly more multifaceted and harder to manage, and this is a tendency that it is going to be more accused with new emerging paradigms of services such as virtualized cloud computing, big data applications, data center services or multimedia content delivery. With the aim of reversing this situation, a symmetric, dynamic, and accessible network architecture, namely software-defined networking (SDN) [1] paradigm, was proposed as a solution to build a more flexible network infrastructure with programmable devices, where new protocols and policies can be implemented via software without needing any hardware modification. The SDN paradigm proposes to separate the control and data planes of "legacy" networks for the sake of flexibility. In this way, the data plane is located in the SDN-enabled forwarding devices (i.e., SDN switches), while the control plane is logically centralized in new entities called SDN controllers. Thus, the different planes in SDN create different layers of abstraction and, thereby, provide an unprecedented level of flexibility. The design of the SDN paradigm enables the performance of a fine-grained management

of the network, taking advantage of decision-making from the global perspective of the network in the controller. However, to be successful in current dynamic environments, traffic monitoring becomes a cornerstone in SDN given that management applications often need to make use of accurate and timely traffic measurements. Forwarding components like switches, routers, etc., are elements of the data plane and the controller is the only element of the control plane. Decoupling of the network control and forwarding functions, and direct programmability of the network give network managers enough control over the network. Separation of the routing and forwarding activities of networking components (e.g., switches, routers, etc.) from the data plane, makes the administration and management of the network straightforward because the control plane now only has to handle logical network topology-related information, traffic routing, and so forth while the data plane only needs to manage the network traffic using the configuration provided by the control unit. The SDN paradigm was the outcome of some works of the research world that suggested the necessity of building programmable networks as a practical way to experiment with new protocols in production networks. From its inception, it has gained lots of attention from academia and industry. It is supported by giants of the Internet world like Google, Cisco, HP, Juniper or NEC and by standardization organizations like the Open Network Foundation (ONF) or the Internet Engineering Task Force (IETF), so one can state that this network paradigm has a lot of potential to succeed. The proposal of the OpenFlow protocol [2] in 2008 was its major driver. In that text, they talk about the commonly held belief that the network infrastructure was "ossified". Thus, they proposed OpenFlow as a protocol for the communication between the forwarding and control planes in order to decouple logically and physically these two planes. OpenFlow [3] allows to dynamically define the forwarding state of the network from the SDN controller by installing in the switches sets of flow entries. These flow entries are stored in flow tables in the switches and determine their behavior. Moreover, the significant contribution of SDN has already been demonstrated by some of the well-established tech giants like Google B4 [4] and Huawei carrier network [5]. Some other examples of SDN controller software that open source communities are currently using include NOX [6], Ryu [7], Beacon [8], Open Daylight [9], and Floodlight [10]. Apart from its advancements in usability, SDN is also associated with various security matters regarding data control and application interface [11]. As security is the key concern for any new paradigm, a collaborative work between academia and industry is significantly imperative in this regard. However, the primary concern of resolving malicious attacks by detecting attack processes rapidly from the network is by using network intrusion detection systems (NIDS). They protect a network from being affected by malicious data. Network intrusion detection systems (NIDS) are developed to detect malicious activities including distributed denial-of-service (DDoS) attacks, virus, worm, and anomaly patterns [12]. A common approach for intrusion detection is detecting anomalies in network traffic, however, network threats are evolving at an unprecedented rate. There are several success factors like anomaly detection speed, accuracy, robustness, and reliability that are considered as the prime success issues for NIDS. Generally, two categories (signature and anomaly-based) of the intrusion detection approach are normally used to protect any network from malicious content. Signature-based detection is the way to detect packets that have a signature in the network traffic corresponding to the rules established in the IDS. Therefore, detecting new and unknown intrusion is not feasible in this type of intrusion detection system. In anomaly-based detection, it will not find attacks using one-to-one correspondence, like signature-based detection; this detection uses the tendency of the attack traffic to determine whether an attack has occurred or not.

Subsequently, new and unidentified intrusion can be identified successfully in the anomaly-based approach. When the anomaly-based intrusion detection system is integrated with flow-based traffic monitoring, which it normally is, it only needs to inspect packet headers. A direct implication is that the flow-based intrusion detection system needs to deal with moderate amount of data. Nowadays in many areas of computer science (CS) and information technology (IT) such as, object detection, natural language processing (NLP), image processing (IP), speech recognition (SR), face detection (FD), machine learning (ML), and deep learning (DL) are being used effectively. Several intrusion

detection methods leveraging machine learning and deep learning are also achieving success gradually. In order to improve detection accuracy and minimize the low false alarm rate, machine learning (ML) techniques are employed to develop NIDS. Due to its advance features other than machine learning, nowadays the deep learning (DL) approach has also been used extensively in the tracks of anomaly detection [12]. This paper will explore the effects of a flow-based anomaly detection system using both Machine learning and deep learning approach in software-defined networking because of the nature of flow-based traffic analysis of software-defined networking (SDN).

We examined the benchmark NSL-KDD dataset for predicting likely attacks types including DoS (denial-of-service) attacks, probe (probing attack), R2L (root to local) attacks, and U2R (user to root) attack. Some of the most efficient classification models including random forest, projective adaptive resonance theory (PART), J48, naïve Bayes (NB), radial basis function network (RBFN), decision tree (DT), and Bayesian network (BN) were applied in our machine learning-based experiment. Apart from this, we also developed a deep neural network model of gated recurrent unit-long short-term memory (GRU-LSTM) for the same dataset to compare the detection accuracy. However, in both cases, for the performance improvement of the classifier, appropriate feature selection methods and pre-processing techniques were employed to ensure the elimination of redundant and irrelevant data from the dataset. Moreover, we designed our methodology in such a way so that we could overcome the shortcomings of the existing solution to this major problem. Several factors influence the detection accuracy of SDN-based intrusion detection systems including appropriate feature selection of dataset, amount of data in dataset, proper choice of ML model, training time, cross validation, learning rate, and so on. Most of the research in literature lacks the detection of all categories of attacks (DoS, Probe, R2L, U2R) in SDN. On the other hand, these existing works have some major shortcomings as they evaluated their performance of the model for a specific ML technique for a specific attack category. However, we have employed both the machine learning and deep learning approach for all sort of attack categories to build a network intrusion detection system and evaluate their performance with the support of the NSL-KDD dataset in order to measure the comparative strengths and weaknesses of both approaches.

The remainder of this paper is organized as follows: Section 2 briefly discusses some of the relevant work as well as a little background to each of these methods. Section 3 provides an overview of the methodology of our research article. We also review both approaches of ML and DL based NIDS implementation model. Furthermore, experimental results and research challenges associated with applying ML/DL to SDN-based NIDS are discussed. Finally, in Section 5 we conclude the paper with future works.

2. Background and Related Work

This section focuses on highlighting the related work for network intrusion detection in SDN with deep learning and machine learning algorithms. In recent years, flow-based anomaly detection classifications have been extensively investigated. Flow-based anomaly detection system, using a multi-layer perceptron (MLP) neural network with one hidden layer and gravitational search algorithm (GSA) proposed in [13], can classify benign and malicious flows with a high degree of accuracy. In [14], the authors introduced a novel concept for an inductive NIDS that uses a one-class support vector machine (SVM) for analysis and is trained with malicious network data in contrast to other systems which give a low false alarm rate. However, the number of traffic anomaly detection algorithms was proposed by authors in [15] by using NOX and OF compliant switches. These algorithms are more successful in detecting anomalies in a small office network but not in an ISP. A lightweight method for DDoS attack detection based on traffic flow features is presented in [16], in which such information is extracted with a very low overhead by taking advantage of a programmatic interface provided by the NOX platform. This method produces a high rate of detection obtained by flow analysis using self-organizing maps (SOM). Kokila et al. [17] analyzed the SVM classifier for DDoS detection and their experiments showed that the SVM classifier gives less false positive rates and high classification accuracy as compared to other techniques. In another research, Trung et al. [18]

proposed an optimized protection mechanism (OpenFlowSIA) that uses SVM classifier along with the authors' proposed idle—timeout adjustment (IA) algorithm to secure and save the network resources under flooding attacks in SDN. A lightweight solution proposed by authors in [19] can detect DDoS attacks within the first 250 malicious traffic packets using the entropy variation of the destination IP address. A DL model based on stacked autoencoder (SAE) developed by Niyaz et al. [20] can detect DDoS multi-vector attacks in SDN. In 2016, Tang et al. [21] applied a deep learning approach to the detection of flow-based anomalies in an SDN environment and developed a deep neural network (DNN) model for an intrusion detection system and trained the model with the NSL-KDD dataset. However, they used only six basic features of the NSL-KDD dataset and surprisingly no appropriate feature selection method was employed. In the later phase Tang et al. [22] presented a gated recurrent unit recurrent neural network (GRU-RNN) which enabled the intrusion detection model and achieved a high degree of accuracy from his previous work; yet still the number of features used was limited to only six. Another approach of developing software-defined network (SDN) controller rules during a DDoS attack was presented by Sen et al. [23]. The authors developed a virtual network testbed based on the SDN environment to implement a DDoS attack and detect that attack leveraging the machine learning model. In addition, they also prepared a network traffic dataset based on SDN to build and test their model of machine learning. A two-level machine-learning intrusion detection system in SDN was proposed by Vetriselvi et al. [24]. They built an IDS by merging the principal of machine learning and genetic algorithms and dividing it into two phases: the former to detect attacks and the latter to categorize them. A detailed study of various approaches based on classical ML techniques that were normally used in detecting attacks in SDN was provided by Elsayed et al. [25]. They performed their benchmarking experiments on the NSL-KDD dataset and explained the consequence of using traditional machine-learning based methods.

However, in very recent times two different approaches of the feature selection-based intrusion detection model proposed by Dey in [26,27] were employed using both deep learning and machine learning approaches for finding higher accuracy in terms of intrusion detection. Apart from that, Dey [28] also showed the performance analysis of SDN-based intrusion detection model for various machine leaning based classifiers with different feature selection approaches.

3. Materials and Methods

In this section of the article, we will briefly discuss our research methodology. We will start with machine learning-based classification model approach with different feature selection methods and the evaluation procedure. Deep learning-based model development will also be discussed in this section in order to capture a translucent comparative idea regarding these two approaches.

3.1. Proposed Classification Model of Machine Learning

Here, we represent our designed machine learning (ML)-based model. A two-layer-based hybrid classification architecture is shown in Figure 1. The top layer, based on certain influential selection methods of features, removes unrelated and inconsistent features and provides the selected features to the next layer. The next layer then uses some fruitful and mostly symmetrical machine learning algorithms to categorize the abridged dataset. Then, the model is further trained and tested using a 10-fold technique of cross-validation. Furthermore, we used various accurate measures to evaluate model performance.

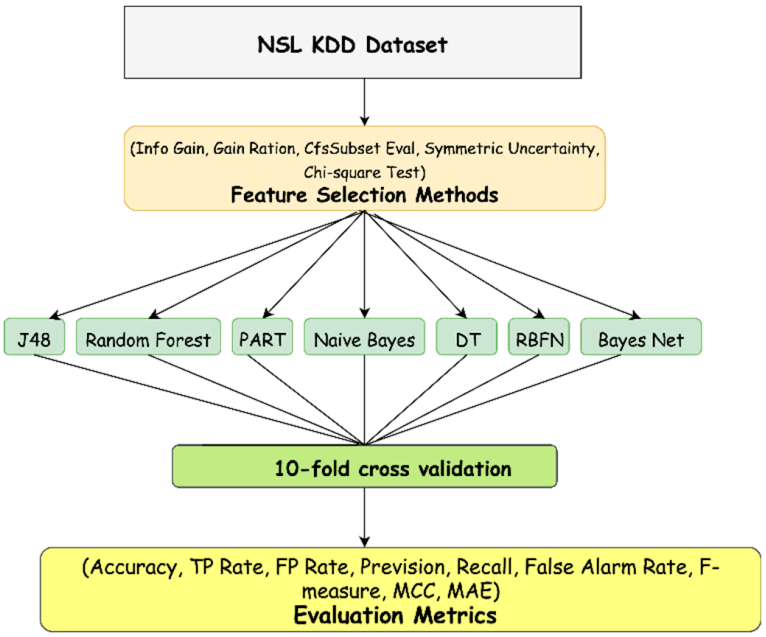

Figure 1. Machine learning based multi-layer classification model [27]. Abbreviations: NSL-KDD; PART: projective adaptive resonance theory; DT: decision tree; RBFN: radial basis function network; TP: true positive; FP: false positive; MCC: Matthews correlation coefficient; MAE: mean absolute error. Reprint with permission [27]; 2019, IEEE.

3.2. Machine Learning Approach

According to [29] the general idea behind most of the machine learning framework is that by training using an example set of training data, the program learns to perform a task. This training enables the distributed computer and controller system to perform similar tasks where the system is confronted with entirely new datasets that have not been previously encountered. Therefore, machine learning can be used for flow-based anomaly detection system to automatically build a predictive model based on the training dataset. To solve numerous classification and prediction problems machine learning algorithms are used [30]. A complete flow chart of the anomaly detection mechanism in OpenFlow controller is shown in Figure 2.

3.3. Overview of NSL-KDD Dataset

It is by no means straightforward to select a precise dataset to establish and evaluate a model. Our aim was to select a small, noise-free, accurate, symmetrical, and redundancy-free set of data. In the past, countless forms of datasets have been used for different anomaly detection systems; some are self-made while others are accessible to the public. Among other things, KDD-99 is a publicly accessible dataset that is most commonly used and embraced. According to the authors of [31] KDD-99 is essentially free to download, but the entire database is massive enough to increase the cost of intrusion detection computing intensively. In contrast, the authors of [32] show that usually only 10% of the dataset is used. However, there is plenty of extraneous information in the training dataset to which there are similar records in the test dataset. As a consequence, the system's learning process is sometimes interrupted. Nevertheless, NSL-KDD is considered a simplified version of KDD-99 that eliminates redundancies between the training dataset and test dataset. Tavallace et al. [31] proposed the flow-based anomaly detection NSL-KDD dataset for intrusion detection. Moreover, for their own research projects, most researchers use the dataset of NSL-KDD. In NSL-KDD, there are 41 features that include both standard patterns and patterns of attack. Table 1 summarizes the overall functionality of the NSL-KDD dataset.

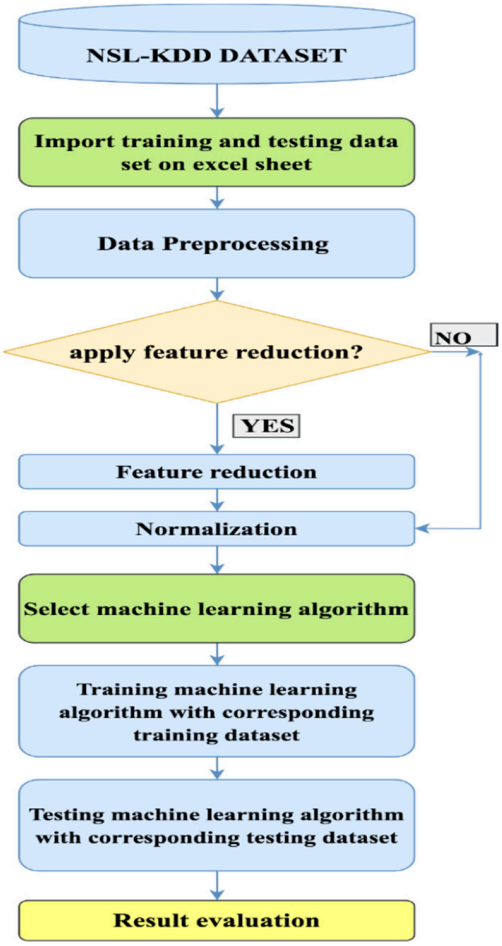

Figure 2. Flow diagram of anomaly detection using the machine learning approach [27]. Reprint with permission [27]; 2019, IEEE.

Table 1. Features and attributes position of NSL-KDD dataset [27]. Reprint with permission [27]; 2019, IEEE.

Type	Features	Attributes Position
Nominal	protocol type, service and flag	2, 3, 4
Numeric	duration, src bytes, st bytes, wrong fragment, urgent, hot, num failed logins, num compromised, num root, num file creations, num shells, num access files, num outbound cmds, count srv count, serror rate, srv serror rate, rerror rate, srv rerror rate, same srv rate, diff srv rate, srv diff host rate, dst host count, dst host srv count, dst host same srv rate, dst host diff srv rate, dst host same src port rate, dst host srv diff host rate, dst host serror rate, dst host srv serror rate, dst host rerror rate and dst host srv rerror rate	1, 5, 6, 8, 9, 10, 11, 13, 16, 17, 18, 19, 20, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41
Binary	land, logged in, root shell, su attempted, is host login and is guest login	7, 12, 14, 15, 21, 22

3.4. Selection of Features for Machine Learning Approach

In the presence of redundant features and attributes in the intrusion dataset, the reliability of anomaly detection decreases. Therefore, it can be a research task to establish methods for the correct selection of features that can sort out dissimilar attributes. Selection of features is the process used

to pick appropriate features and delete irrelevant features from the dataset to complete a given task [33]. In addition, reducing the number of redundant, unnecessary, and noisy features will lead to an improved design by accelerating a data-mining algorithm with an improvement in accuracy of learning [34]. Using info gain, gain ratio, CFS subset evaluation, symmetric uncertainty, and the Chi-squared test, we refined NSL-KDD dataset features. Table 2 displays attribute collection procedures for different evaluators and their different search strategies.

Table 2. Feature selection with different evaluator and search method [27]. Reprint with permission [27]; 2019, IEEE.

Evaluator	Search	Selected Attributes
Info Gain	Ranker	5,6,3,4,33,35,34,40,41,23,30,29,12,27,28:15
Gain Ratio	Ranker	28,12,41,27,4,6,5,30,29,40,3,25,26,39,34:15
CFS Subset Evaluator	Best First	5,6,12,25,28,30,31,37,41:9
Symmetric Uncertainty	Ranker	6,5,4,41,28,12,27,30,3,40,29,34,35,33,37:15
Chi-Squared Test	Ranker	5,6,3,33,35,34,4,40,23,12,41,30,29,27,37:15

3.5. Random Forest Classifier (RF)

For supervised learning tasks, the random forest classifier is known as an ensemble machine learning technique. Each individual tree spouts a class prediction in the random forest, and the class with the most votes becomes the prediction of the system. Breiman initially suggested this algorithm in which the author identified the benefits of random forest [35]. There are several advantages for which it produces higher accuracy in model classification. Figure 3 shows how the random forest algorithm works with the dataset. Initially, it follows four steps: firstly, it selects random samples from a given dataset. It then creates a decision tree for each test and gets a predictive result and then holds a vote for each predicted outcome. Ultimately, with the most votes, it picks the predictive outcome as the final forecast.

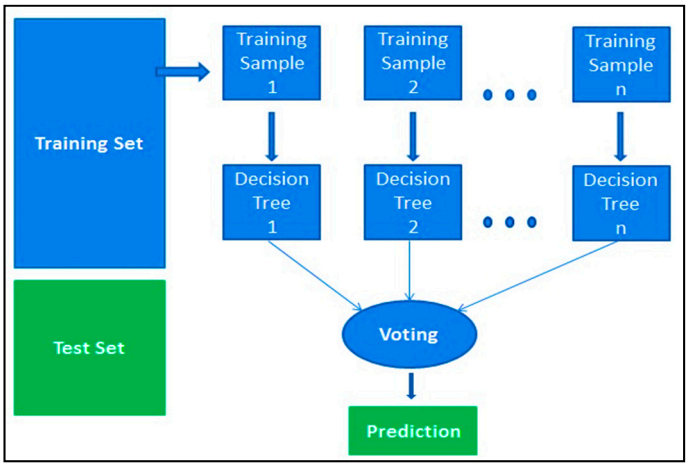

Figure 3. Working procedure of random forest algorithm.

3.6. Evaluation Metrics

Accuracy (AC), precision (P), recall (R), F-measure (F), false alarm rate (FAR), and Matthews correlation coefficient (MCC) measure the performance of the intrusion detection rate taking into account the performance metrics derived from the confusion matrix to calculate the value of these evaluation metrics. The confusion matrix illustrates the performance of the algorithm according to Table 3.

Table 3. Tabular form of a confusion matrix.

	Predicted as Normal	**Predicted as Attack**
Normal Class (Actually)	True Positive (TP)	False Positive (FP)
Attack Class (Actually)	False Negative (FN)	True Negative (TN)

A good intrusion detection scheme involves a high rate of accuracy and high detection rate with a very low false alarm rate. The false alarm rate is directly proportional to the miss classification rate. A brief discussion and calculating formula 170 for the metrics that are used to evaluate the model are given below:

Accuracy (AC): Shows the category proportion over all N cases that were correct.

$$AC = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision (P): Demonstrates the percentage of intrusion detection system in the network that detects intrusion that is intrusion. The higher the value of P, the lower the rate of false alarm.

$$P = \frac{TP}{TP + FP}$$

Recall (R): Illustrates the proportion of positive examples properly classified. We are looking for a high R value.

$$R = \frac{TP}{TP + FN}$$

F-measure (F): It provides an improved measure of accuracy by providing a balance between accuracy and recall. We are looking for a high F-measure value.

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Matthews correlation coefficient (MCC): It returns a binary value of −1 to 1.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

3.7. Deep Learning Approach

In this segment, we will briefly discuss our proposed network intrusion detection system for the combined gated recurrent unit-long short-term memory (GRU-LSTM). An appropriate ANOVA F-test and recursive feature elimination (RFE) (ANOVA F-RFE) selection method was also applied to improve classifier performance. Moreover, in this section we will also discuss RNN, GRU, LSTM, ANOVA F-RFE method, and the proposed algorithm for detecting attack patterns in a DL model.

3.7.1. Recurrent Neural Network (RNN)

Recurrent neural network (RNN) [36] is a type of artificial neural network which has the capability of learning from previous time-steps. RNNs are extended forms of typical feed-forward neural networks (FNNs) [37]. But, in contrast with FNNs, RNNs use their internal state while processing sequential data. Using the internal state, here, refers to the fact that, the RNN takes advantage of previous computations for output. As they carry out the same task for each element in the sequence, they are called recurrent. The structure of a simple RNN is depicted in Figure 4.

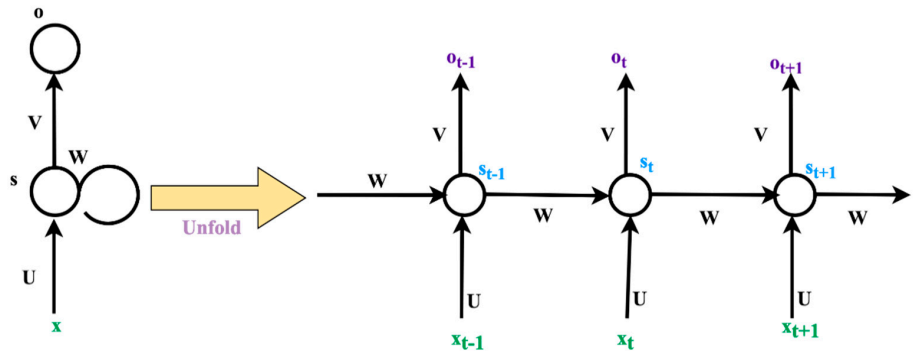

Figure 4. Structure of a plain recurrent neural network (RNN) [26]. Reprint with permission [26]; 2019, IEEE.

In the above figure, x_t is input and o_t is output; s_t is considered as hidden state; f indicates nonlinear function, such as *tanh* or *ReLU*; s_t is calculated using the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$. To calculate the initial hidden state, s_{-1}, is required which is initialized 185 to zero by default. Hidden sates of RNN are computed as:

$$S_t = f\left(U_{x_t} + W_{S_{t-1}}\right),\ for\ t = T, \ldots, 1 \tag{1}$$

A gradient-based algorithm, namely backpropagation through time (BPTT), is generally applied for training the RNN. RNN training is considerably faster using BPTT algorithm than other existing optimization techniques. However, the RNN model with backpropagation has a significant drawback, called the vanishing gradient problem. It happens when the gradient is so small that it seems vanished. Consequently, it prevents the value of weight from changing, and in some cases, stops further training. According to [38], the vanishing gradient problem prevents the RNN from being accurate. To solve these problems, more powerful combined models like long short-term memory (LSTM) [39] and gated recurrent units (GRUs) [40] were suggested.

3.7.2. Long Short-Term Memory (LSTM) RNN

A deep neural network was developed in time and an FNN was built for each time-step. Then, weights and biases for each hidden layer were updated by the gradient rule. These updates minimize the loss between the expected and actual outputs. But, when the time-steps are more than 5–10, standard RNNs do not perform better. Weights fluctuate due to the prolonged back-propagation vanishing or blowing up error signals, making the network performance poor. Accordingly, researchers suggested the long short-term memory (LSTM) network to address this fading gradient problem. LSTM bridges the time gaps and uses a gating mechanism to deal with long-term dependencies. Figure 5 shows the LSTM structure.

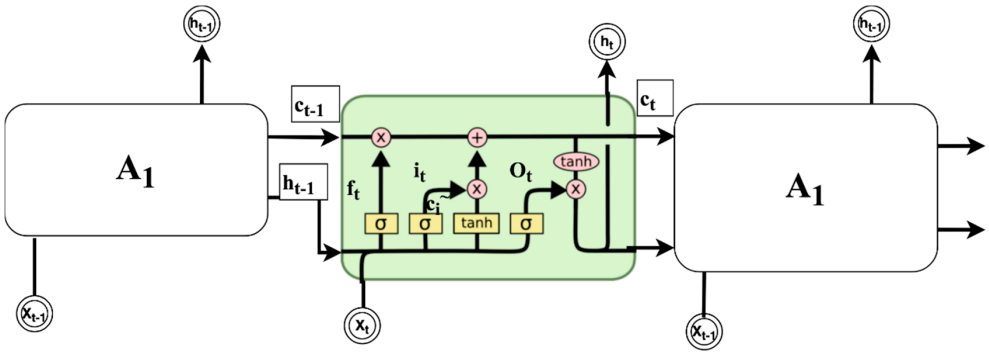

Figure 5. Simple structure of a long short-term memory (LSTM) unit [26]. Reprint with permission [26]; 2019, IEEE.

3.7.3. Gated Recurrent Unit (GRU)

A gated recurrent unit (GRU) is an LSTM's lighter version. The reduced complexity in a GRU's structure is achieved by decreasing the architectural gates. GRU uses both the update gate and the reset gate to solve the vanishing gradient problem of a regular RNN. In essence, these are two vectors that determine what information should be passed on to the output. Since a GRU's training phase is smoother and faster than LSTM, we chose GRU to develop our model [41]. Both the "forget gate" and "input gate" in an LSTM are merged into an "update gate" in GRU and the hidden state and cell state are combined, resulting in a simpler structure as shown in Figure 6.

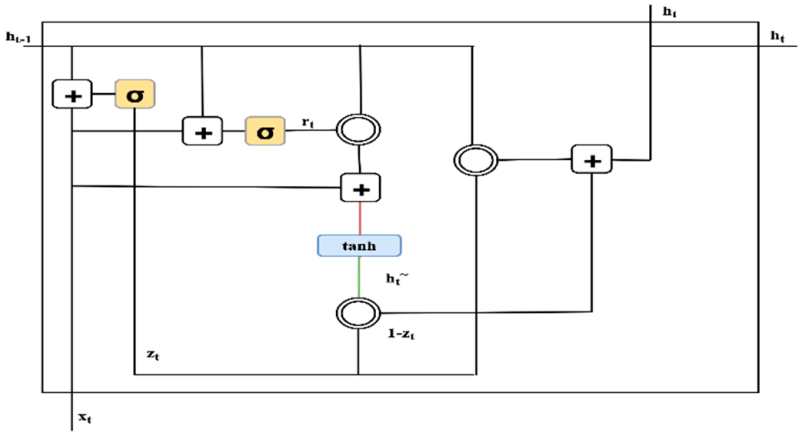

Figure 6. Single layer gated recurrent unit (GRU) [26]. Reprint with permission [26]; 2019, IEEE.

The following relationship can be obtained from Figure 6.

$$\text{Update gate } z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right) \tag{2}$$

$$\text{Candidate activation } \widetilde{h_t} = \tan\mathrm{h}(Wx_t + r_t \odot Uh_{t-1}) \tag{3}$$

$$\text{Reset gate } r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1} \tag{4}$$

$$\text{Activation function } h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{5}$$

3.7.4. Multi-Layer GRU RNN

The performance of the algorithm depends heavily on the numerous deep neural network architectures. A deep structure called multi-layer RNN is designed stacking various RNN layers (plain RNN, LSTM, GRU). In each hidden layer, an RNN that uses GRU cells is called GRU-RNN. In addition to back-propagation through time, in the multilayer structure, network input is passed through multiple GRU layers. In [42], multilayered RNNs have been shown to learn from the various time lengths of input sequences. Multi-layered RNNs share the hyper parameters, weights, and biases across the layers to achieve optimized efficiency.

3.7.5. Overview of Scikit-Learn

During experiment, we used scikit-learn which is a python-based machine learning library for data mining and data analysis [43]. Most machine learning algorithm data must be stored in either a two-dimensional (2D) array or matrix form. Such 2D form data can be effectively processed in scikit-learn. Figure 7 shows the representation of the scikit-learn data, where N samples and D features are present.

$$feature\ matrix: x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ x_{31} & x_{32} & \dots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}$$

$$label\ vector: y = [y_1, y_2, y_3, \dots, y_N]$$

Figure 7. Data representation in scikit-learn.

3.7.6. Appropriate Feature Selection for Deep Learning Approach

Feature selection mechanism is a required process to get rid of the irrelevant and extraneous data from the dataset. According to [43], feature selection is a process of deriving a subset of relevant features from the complete feature set without decaying presentation. An intrusion dataset containing superfluous attributes often prevents detection from being accurate. Numerous reasons were analyzed to show why restricting the features is obligatory. Irrelevant features increase computation time without contributing to classifier improvement and sometimes incorrectly indicate correlation between feature and desired class. In our experiment, we have used a univariate feature selection with analysis of variance (ANOVA) F-test. ANOVA is used to determine whether the means of some groups are different using the F-test which statistically checks the equality of means. Each feature is individually analyzed which calculates the strength of feature–labels relationship. The percentile of the highest scores-based feature selection is performed by the SelectPercentile method (sklearn.feature selection). Upon finding a subset, recursive feature elimination (RFE) is applied. RFE also builds a model where features are left aside, and it repeats the process until all features are eliminated in the dataset. By using the weight of a classifier, feature ranking is developed. After applying both ANOVA F-test and RFE, the selected features are shown in Table 4. The attack groups in the NSL-KDD dataset are divided into four types, namely DoS, Probe, R2L, and U2R, according to Table 4.

- DoS: Denial-of-service is considered a major category of attack which reduces the capacity of the victim, thereby rendering it unable to handle valid requests. Syn flooding is an example of a DoS attack.
- Probing: In this process, attackers gain information about the remote victim by surveillance and other probing attacks like port scanning.
- U2R: Unauthorized access to local super user (root) privileges is a type of attack by which an attacker logs into a victim system using a standard account and tries to obtain root/admin privileges by exploiting some vulnerability in the victim.
- R2L: Unauthorized access from a remote machine, the attacker enters a remote machine and gains the local access of the victim's machine. For example, the guessing of the password.

Table 4. Selected features after applying analysis of variance (ANOVA) F-test and recursive feature elimination (RFE) [26]. Reprint with permission [26]; 2019, IEEE.

Attack Category	Selected Features
Denial-of-Service (DoS)	(1, 'flag SF'), (2, 'dst host serror rate'), (3, 'same srv rate'), (4, 'count'), (5, 'dst host srv count'), (6,'dst host same srv rate'), (7, 'logged in'), (8, 'dst host count'), (9, 'serror rate'), (10, 'dst host srv serror rate'), (11,'srv serror rate'), (12, 'service http'), (13, 'flag S0')
Probe	(1, 'service private'), (2, 'service eco i'), (3,dst host srv count'), (4, 'dst host same src port rate'), (5, 'dst host srv rerror rate'), (6, 'dst host diff srv rate'), (7, 'dst host srv diff host rate'), (8, 'dst host rerror rate'), (9, 'logged in'), (10, 'srv rerror rate'), (11,'Protocol type icmp'), (12, 'rerror rate'), (13, 'flag SF')
Root to Local (R2L)	(1, 'src bytes'), (2, 'hot'), (3, 'dst host same src port rate'), (4,'dst host srv count'), (5, 'dst host srv diff host rate'), (6, 'dst bytes'), (7, 'service ftp data'), (8, 'num failed logins'), (9, 'is guest login'), (10, 'service imap4'), (11, 'service ftp'), (12,'flag RSTO'), (13, 'service http')
User to Root (U2R)	(1, 'hot'), (2, 'dst host srv count'), (3, 'dst host count'), (4,'num file creations'), (5,'root shell'),(6,'dst host same src port rate'),(7,'dst host srv diff host rate'), (8, 'service ftp data'), (9,'service telnet'), (10, 'num shells'), (11, 'urgent'), (12,'service http'), (13, 'srv diff host rate')

3.8. Designed Algorithm and Proposed SDN-Based Anomaly Detection Architecture

Normally the controller unit's OpenFlow switches are managed by the SDN controller. Whenever required, the SDN controller is able to request all network data. Therefore, for both machine learning and deep learning methods, we implemented our proposed section of intrusion detection in the SDN controller, as illustrated in Figure 8. Our suggested approach for the ML-based classification model is summarized in the following algorithm.

Algorithm 1: Machine learning-based anomaly class detector for software-defined networking (SDN) attacks

```
1  PROCEDURE: FLOW BASED DETECTOR BASED ON ML MODEL
2      Selection of appropriate machine learning algorithm;
3      Train the ML-based model using benchmark dataset NSL-KDD;
4      Nomination of appropriate feature after using different feature selection methods;
5      if the trained model predicts an anomaly class on an OpenFlow
           Controller by the M- based intrusion detection model then
6          Update the SDN OpenFlow controller rules to block that class
               attack type;
7      else
8          Allow the normal class to pass through SDN controller and access
           the available resources;
9      end
10 end
```

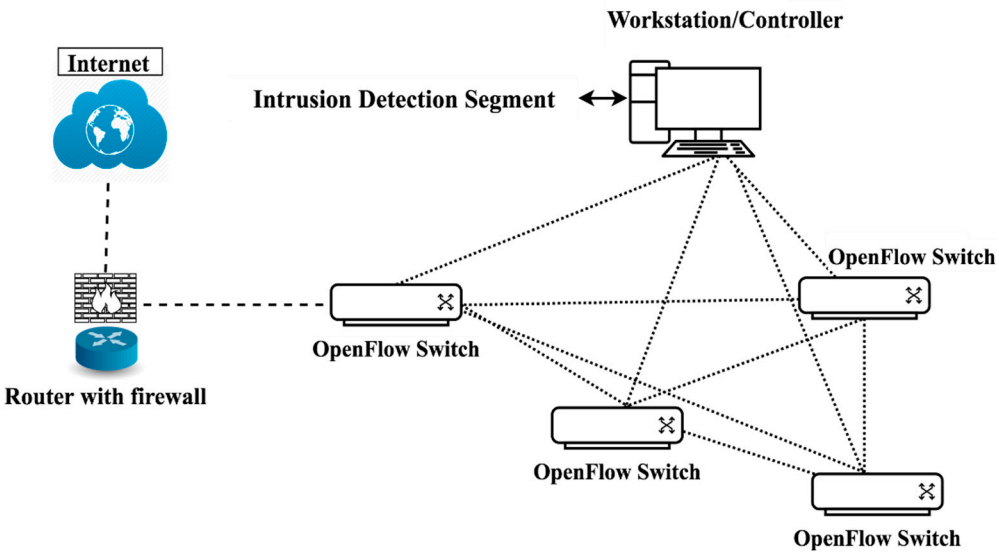

Figure 8. Proposed flow-based anomaly detection architecture in SDN [28]. Reprint with permission [28]; 2019, Springer.

To request network data, a request message for OpenFlow stats will be sent to all OpenFlow switches from the controller. An OpenFlow stats reply message with all available data is sent back to the controller by the OpenFlow switch as a controller request for all available statistics. Figure 9 clearly describes how the OpenFlow switch handles the incoming packet and responds by using the Open Flow protocol according to the availability of data in the flow table. One of the SDN's noticeable behaviors is that its centralized controller can take full network opportunities to assess and associate network feedback. Thus, when a network anomaly is discovered and recognized, the OpenFlow protocol can effectively alleviate an intrusion via flow table adjustment. Algorithm 2 summarizes our suggested solution to the deep learning-based classification system.

Algorithm 2: Deep learning-based anomaly class detector for SDN attacks.

```
1  PROCEDURE: FLOW BASED DETECTOR BASED ON DEEP MODEL
2      Selection of appropriate deep learning algorithm;
3      Train the DNN-based model using benchmark dataset NSL-KDD;
4        Nomination of appropriate feature after using different feature selection methods;
5      if the trained model predicts an anomaly class on an OpenFlow
         Controller by the DNN-based intrusion detection model then
6          Update the SDN OpenFlow controller rules to block that class
             attack type;
7      else
8          Allow the normal class to pass through SDN controller and access
           the available resources;
9      end
10 end
```

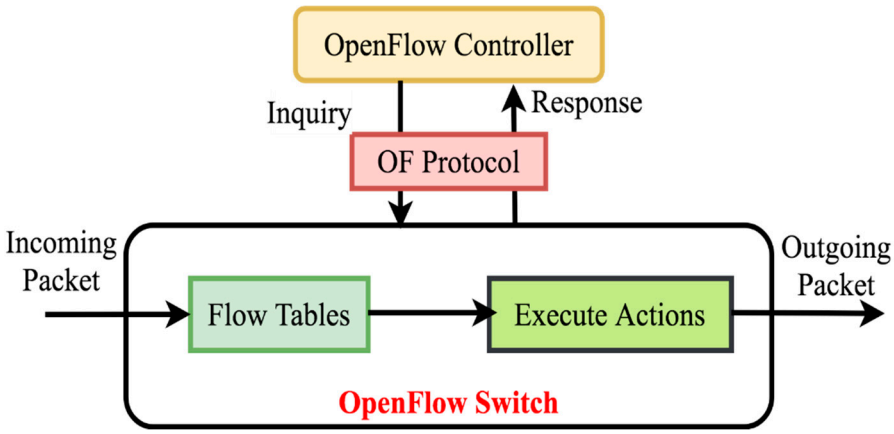

Figure 9. Diagram of handling incoming packets in OpenFlow switch [26]. Reprint with permission [26]; 2019, IEEE.

4. Experimental Results

4.1. Experimental Results of Machine Learning Approach

To carry out the experiments, we used the WEKA [44] environment and the NSL-KDD dataset. The system consists of a 6 GB hard processor and an Intel(R) Core(TM) Processor(s) i5-2410 M CPU @ 2.30 GHz, 2301 MHz, Dual Core(s) and four Logical Processor(s). WEKA's trouble-free heap size was increased to load and analyze the dataset. The NSL-KDD dataset was used to train and test each of the selected 285 features. We used a 10-fold cross-validation approach in our experiment to successfully conduct the experiment. By dividing the training set into 10 subsets, we tested each subset when the model was trained on the other nine subsets. Each subset, however, is processed as test data only once; the process therefore repeats up to 10 times. For simplicity, only results of the higher accuracy classifier obtained with different method of selection of features were mentioned in Table 5.

Table 5. Random forest classifier showing the highest accuracy with gain ratio feature selection.

Feature Selection Method	Classifier Techniques	Accuracy	TP Rate	FP Rate	Evaluation Criteria			F-Measure	MCC	MAE
					Precision	Recall	FAR			
Info Gain	Random Forest	79.360	0.794	0.163	0.846	0.794	0.341	0.792	0.641	0.229
CFS Subset	PART	79.249	0.792	0.167	0.839	0.792	0.333	0.791	0.633	0.264
Gain Ratio	**Random Forest**	**81.946**	**0.819**	**0.143**	**0.860**	**0.819**	**0.297**	**0.819**	**0.681**	**0.232**
Symmetric Uncertainty	Random Forest	80.708	0.807	0.153	0.853	0.807	0.317	0.806	0.661	0.221
Chi-squared	Random Forest	80.132	0.801	0.157	0.850	0.801	0.328	0.800	0.653	0.222

Table 6 represents the results of an entire classifier with different feature selection method. Different results of accuracy (AC), true positive rate (TPR), false positive rate (FPR), precision (P), recall (R), false alarm rate (FAR), F-measure, Matthews correlation coefficient (MCC), and mean absolute error (MAE) of specific classifiers are shown based on the test. The Table uses different color combinations to understand the highest accuracy of specific classifier techniques with numerous methods of features selection. In addition, only the highest results obtained from different classifiers are shown in terms of the selection of features. The best performance accuracy for each individual feature selection methods is indicated by each colored classifier technique. The following colored relation of IG-RF, CFS-PART, GR-RF, SU-RF, CST-RF shows the best classifier methods in terms of higher accuracy and lower false alarm rate. Our initial goal, however, was to achieve high accuracy, recall, and low false alarm frequency and MCC. We accomplished that successfully in our experiment. From the experimental data, random forest with gain ratio feature selection method exhibits the highest accuracy of 81.946% which is illustrated from our experimental results. We plotted the accuracy value of ML-based models in Figures 10 and 11. After analyzing these results, we found that the info gain feature selection approach

produced a higher accuracy of 79.36% with random forest classifier whereas PART shows an accuracy of 79.249% with CFS subset evaluator. However, the rest of the feature selection methods illustrate an accuracy value of more than 80% with RF and PART, respectively. Among all of them, random forest with gain ratio selection method produced the highest accuracy of nearly 82% (~81.946%).

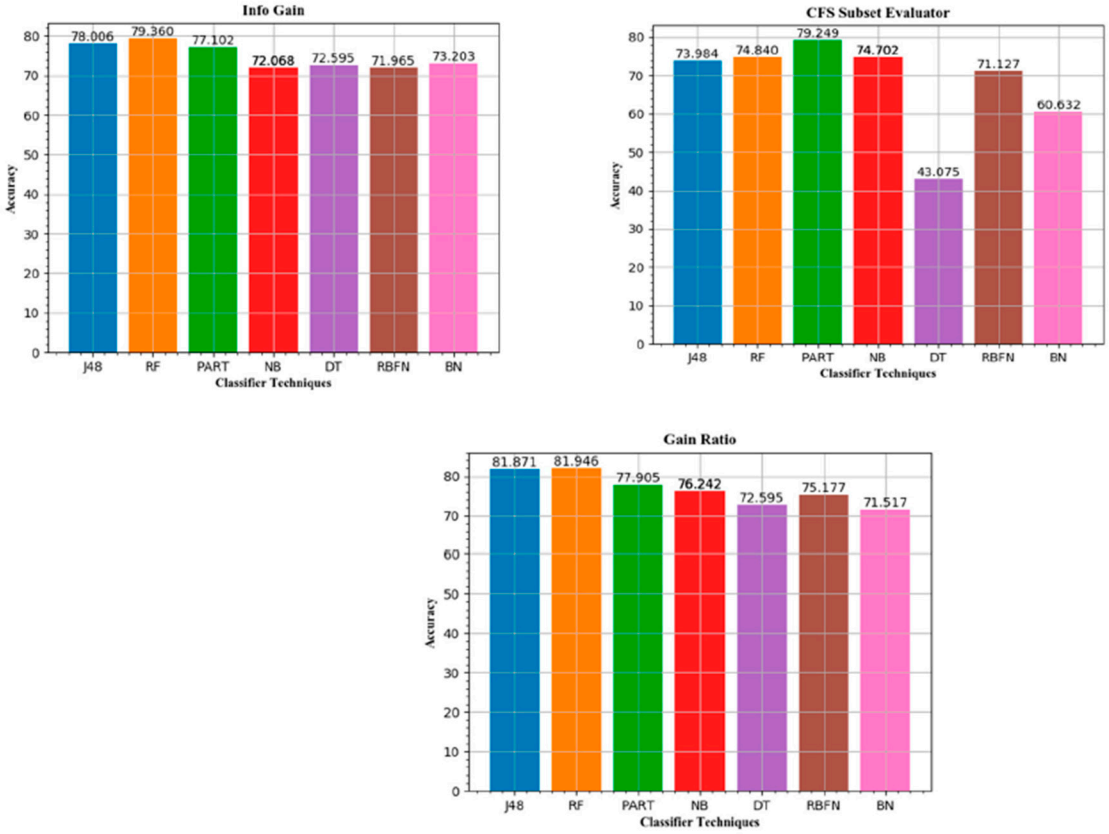

Figure 10. Accuracy value of machine learning (ML)-based detection model for info gain, CFS subset, and gain ratio for all classifier techniques.

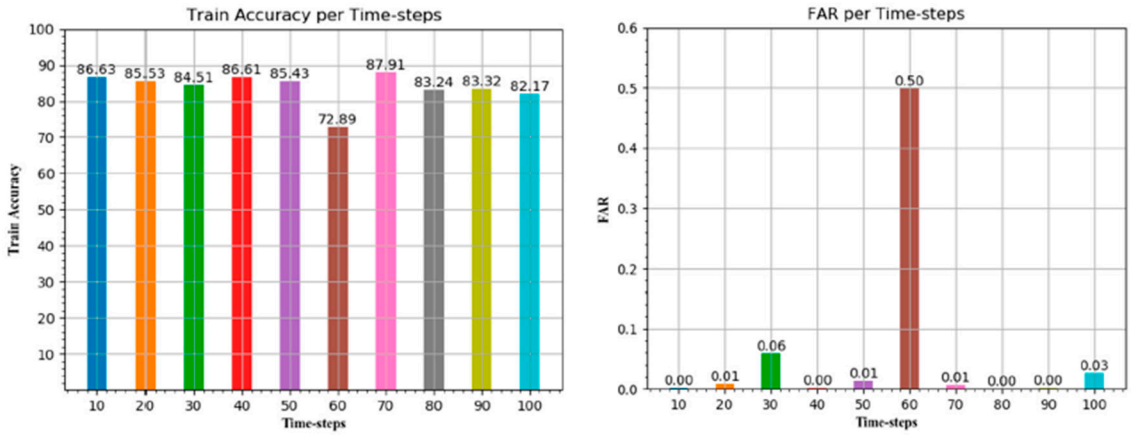

Figure 11. Accuracy value of ML-based detection model for symmetric uncertainty and Chi-squared test for all classifier techniques.

Table 6. Results of all classifiers with different feature selection method.

Feature Selection Method	Classifier Techniques	Evaluation Criteria								
		Accuracy	TP Rate	FP Rate	Precision	Recall	FAR	F-Measure	MCC	MAE
Info Gain	J48	78.006	0.781	0.172	0.84	0.781	0.364	0.779	0.623	0.229
	Random Forest	**79.360**	**0.794**	**0.163**	**0.846**	**0.794**	**0.341**	**0.792**	**0.641**	**0.229**
	PART	77.102	0.771	0.18	0.835	0.771	0.382	0.768	0.609	0.231
	Naive Bayes	72.068	0.721	0.227	0.789	0.721	0.442	0.715	0.514	0.279
	DT	72.595	0.726	0.214	0.814	0.726	0.461	0.718	0.545	0.197
	RBFN	71.965	0.72	0.228	0.787	0.72	0.441	0.714	0.511	0.299
	Bayes Net	73.203	0.732	0.209	0.816	0.732	0.45	0.725	0.553	0.268
CFS Subset Evaluator	J48	73.984	0.740	0. 203	0.820	0.74	0.436	0.734	0.564	0.267
	Random Forest	74.84	0.784	0.197	0.823	0.748	0.42	0.743	0.575	0.345
	PART	**79.249**	**0.792**	**0.167**	**0.839**	**0.792**	**0.333**	**0.791**	**0.633**	**0.264**
	Naive Bayes	74.702	0.747	0.829	0.829	0.747	0.43	0.741	0.58	0.253
	DT	43.075	0.431	0.431	0.186	0.431	1	0.259	0	0.504
	RBFN	71.127	0.711	0.222	0.817	0.711	0.505	0.7	0.533	0.323
	Bayes Net	60.632	0.606	0.298	0.794	0.606	0.691	0.564	0.401	0.447
Gain Ratio	J48	81.871	0.819	0.145	0.858	0.819	0.293	0.818	0.677	0.193
	Random Forest	**81.946**	**0.819**	**0.143**	**0.860**	**0.819**	**0.297**	**0.819**	**0.681**	**0.232**
	PART	77.905	0.779	0.179	0.835	0.779	0.362	0.777	0.616	0.231
	Naive Bayes	76.242	0.762	0.186	0.832	0.762	0.398	0.758	0.597	0.237
	DT	72.595	0.726	0.214	0.814	0.726	0.461	0.718	0.545	0.197
	RBFN	75.177	0.752	0.193	0.828	0.752	0.419	0.747	0.584	0.272
	Bayes Net	71.517	0.715	0.221	0.812	0.715	0.483	0.705	0.532	—
Symmetric Uncertainty	J48	78.927	0.789	0.167	0.842	0.789	0.346	0.787	0.633	0.218
	Random Forest	**80.708**	**0.807**	**0.153**	**0.853**	**0.807**	**0.317**	**0.806**	**0.661**	**0.221**
	PART	80.371	0.804	0.157	0.848	0.804	0.318	0.803	0.653	0.221
	Naive Bayes	73.292	0.733	0.21	0.813	0.733	0.444	0.726	0.551	0.266
	DT	72.595	0.726	0.214	0.814	0.726	0.461	0.718	0.545	0.197
	RBFN	73.522	0.735	0.209	0.812	0.735	0.438	0.729	0.552	0.288
	Bayes Net	71.562	0.716	0.222	0.808	0.716	0.478	0.706	0.529	0.282
Chi-square Test	J48	78.051	0.781	0.173	0.838	0.781	0.363	0.778	0.621	0.229
	Random Forest	**80.132**	**0.801**	**0.157**	**0.850**	**0.801**	**0.328**	**0.800**	**0.653**	**0.222**
	PART	77.989	0.78	0.173	0.84	0.78	0.367	0.777	0.622	0.218
	Naive Bayes	72.618	0.726	0.224	0.79	0.726	0.43	0.722	0.521	0.273
	DT	72.595	0.726	0.214	0.814	0.726	0.461	0.718	0.545	0.197
	RBFN	70.723	0.707	0.234	0.789	0.707	0.475	0.699	0.502	0.31
	Bayes Net	72.409	0.724	0.215	0.812	0.724	0.463	0.716	0.541	0.275

4.2. Experimental Results of Deep Learning Approach

We used Google TensorFlow [45] to carry out the experiments. TensorFlow offers an option for viewing the design of the network. The tests were performed with the Ubuntu 16.10 Distribution Operating System based on Linux in an atmosphere of Intel i5 3.2 GHz, 16 GB RAM, and NVIDIA GTX 1070. We used the tf.train.AdamOptimizer from TensorFlow. Table 7 shows the hyper parameter configuration initialization. The learning rate is controlled by Kingma and Ba's Adam algorithm in tf.train.AdamOptimizer.

Table 7. Set of different hyper parameters.

(a) Hyper parameters
learning rate = 0.001 training epochs = 10 display step = 1 num layers = 1
(b) Definition of hyper parameters for the model
learning rate = 0.001 number of classes = 2 display step = 100 input features = train X.shape [1] #No of selected features training cycles = 1000 #No of time-steps to back propagate time-steps = 5 hidden units = 50 #No of LSTM units in a LSTM hidden layer

The model was developed using Python programming language along with several libraries like python based numpy, machine learning based scikit-learn, pandas for data visualization, and TensorFlow for model development. We started our experiments with a lightweight GRU with a hidden layer and a hidden unit. Ten sets of experiments were performed for each set of hyper parameters (learning speed, time-steps, hidden layers) and we tuned them to get the optimal results. Table 8 represents the results of various evaluation metrics like accuracy, precision, recall, false alarm rate, and F-1 score for each time-step.

Figure 11 illustrates the accuracy values for all time-steps with false alarm rate. At 70 time-steps, deep learning model of GRU-LSTM produces an accuracy of 87.91% (nearly ~88%) which is higher than others.

Table 8. Evaluation metrics for all layer ids classifier. FAR: false alarm rate.

Time-Steps	Train Accuracy	Precision	Recall	F-1 Score	FAR
10	86.632	0.9994	0.99	0.9977	0.0022
20	85.534	0.9943	0.3296	0.9922	0.0077
30	84.510	0.986	0.9952	0.9418	0.05812
40	86.613	0.9996	0.9902	0.9983	0.0016
50	85.434	0.9967	0.9919	0.9865	0.0134
60	72.89	0.8914	0.9935	0.5011	0.4988
70	87.911	0.9981	0.9939	0.9923	0.0076
80	83.243	0.9999	0.9842	0.9997	0.0002
90	83.323	0.9995	0.9859	0.9981	0.0018
100	82.167	0.9937	0.9925	0.974	0.0257

4.3. Comparative Analysis of Two Approaches

In this section, we will briefly discuss the experimental procedures and analyze our results for further use in the division of intrusion detection. Feature selection is considered a prime component of this research work. For both approaches we prepared our dataset by applying some fruitful feature selection algorithms. Researchers from different domains have previously used the NSL-KDD dataset to detect intrusion, but none of the approaches followed a proper selection approach for their experiment. In the machine learning approach, we used some well researched machine learning algorithms like J48, random forest, PART, naïve Bayes, decision tree, radial basis function network (RBFN), and Bayes net. In order to eliminate the ambiguous data from the dataset, we also employed some feature selection algorithms like info gain, gain ratio, CFS subset evaluator, symmetric uncertainty, and Chi-square test. After successful experiments we found that random forest classifier with fain ratio feature selection approach generates 81.946% accuracy with a very low false alarm rate of 0.297%. Apart from that, we also developed a GRU-LSTM-based deep learning model with ANOVA F-test and recursive feature elimination (RFE) selection approach. As our aim is to achieve a high detection accuracy in terms of different approaches with feature selection methods, we tested our model with different time-steps and different learning rates. After successful experiments we observed that with a learning rate of 0.01 and 70 time-steps our model achieved a detection accuracy of approximately 88%. Complete results for 0.01 learning rate with time-steps (10, 20, 30, ... 90, 100) are depicted in Table 8. At this point the deep learning approach shows many potential outcomes compared with the machine learning approach. In our research implementation of the SDN-based intrusion detection system, the classification model is mainly two-class-based, namely normal and anomaly. After evaluating both results, we proposed a model of security architecture which detects flow-based anomaly in an OpenFlow-based controller. From the detailed results, we can derive a decision that, with a very low false alarm rate of 0.0076%, the ANOVA F-Test and recursive feature elimination (RFE) methods with GRU-LSTM classifier provide maximum accuracy of 87.911%. Furthermore, we generated the results using the selected features from the complete NSL-KDD dataset. Some other approaches from different authors were presented

to show the accuracy of the NSL-KDD dataset deep learning algorithm. Nevertheless, there was no pre-processing of the database and the correct choice of features for testing and training.

5. Conclusions

In this research, we have presented two different approaches for predicting the flow-based anomaly in software-defined networking. The GRU-LSTM model based on deep learning and the random forest (RF) model based on machine learning were designed to detect network interference in SDN. In addition, with ANOVA F-Test and RFE feature selection and the gain ratio feature selection method, we also developed the best classifier model in terms of different evaluation metrics. Although both approaches produce significant experimental results compared with other works, both approaches made some effective contribution in the field of intrusion detection for SDN use. It is evident from the experimental results that the deep learning approach produces slightly better results than the machine learning approach, therefore the use of GRU-LSTM model for flow-based anomaly detection is absolutely essential in order to achieve high accuracy and speed up the intrusion detection process in SDN. Nonetheless, we plan to implement our proposed model in the near future in a real SDN environment with real network traffic.

Author Contributions: Conceptualization, S.K.D. and M.M.R.; methodology, S.K.D.; software, S.K.D.; validation, M.M.R.; investigation, S.K.D; resources, S.K.D.; data curation, S.K.D.; writing—original draft preparation, S.K.D; writing—review and editing, M.M.R.; visualization, S.K.D.; supervision, M.M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors are grateful to the Department of Computer Science and Engineering (CSE), Military Institute of Science and Technology (MIST), Mirpur Cantonment, Dhaka-1216 for providing us with the opportunity to carry out this research.

Conflicts of Interest: There were no conflicts of interest with the concerned persons or organizations.

Abbreviations

The following abbreviations are used in the manuscript:

ANOVA	Analysis of variance
BPTT	Backpropagation through time (BPTT)
DDoS	Distributed denial-of-service
DNN	Deep neural network
DoS	Denial-of-service
GRU	Gated recurrent unit
GSA	Gravitational search algorithm
LSTM	Long short-term memory
MLP	Multi-layer perceptron
NIDS	Network intrusion detection systems
OF	Open flow
R2L	Root to local
RFE	Recursive feature elimination
RNN	Recurrent neural network
SAE	Stacked auto encoder
SDN	Software-defined networking
SOHO	Small office/home office
SVM	Support vector machine
U2R	Use to root

References

1. Software Defined Networking Definition. Available online: https://www.opennetworking.org/sdn-definition (accessed on 16 May 2017).
2. ONF SDN Evolution. Available online: http://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/05/TR-535_ONF_SDN_Evolution.pdf (accessed on 25 February 2018).
3. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
4. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined wan. *SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 3–14. [CrossRef]
5. C.t. Huawei Press Centre and H. Unveil World's First Commercial Deployment of SDN in Carrier Networks. Available online: http::://pr.huawei.com/en/news/hw-332209-sdn.htm (accessed on 28 February 2018).
6. Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 105–110. [CrossRef]
7. Ryu. Available online: http://osrg.github.io/ryu (accessed on 11 March 2018).
8. Erickson, D. The beacon openflow controller. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, (HotSDN '13), Hong Kong, China, 16 August 2013; ACM: New York, NY, USA; pp. 13–18. [CrossRef]
9. Opendaylight: A Linux Foundation Collaborative Project. Available online: http://www.opendaylight.org (accessed on 6 March 2018).
10. Floodlight. Available online: http://www.projectfloodlight.org (accessed on 15 March 2018).
11. Kreutz, D.; Ramos, F.M.; Verissimo, P. Towards Secure and Dependable Software-Defined Networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, (HotSDN '13), Hong Kong, China, 16 August 2013; ACM: New York, NY, USA; pp. 55–60. [CrossRef]
12. Sultana, N.; Chilamkurti, N.; Peng, W.; Alhadad, R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 493. [CrossRef]
13. Jadidi, Z.; Muthukkumarasamy, V.; Sithirasenan, E.; Sheikhan, M. Flow-Based Anomaly Detection Using Neural Network Optimized with Gsa Algorithm. In Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, Philadelphia, PA, USA, 8–11 July 2013; pp. 76–81. [CrossRef]
14. Winter, P.; Hermann, E.; Zeilinger, M. Inductive Intrusion Detection in Flow-Based Network Data Using One-Class Support Vector Machines. In Proceedings of the 2011 4th IFIP International Conference on New Technologies, Mobility and Security, Paris, France, 7–10 February 2011; pp. 1–5. [CrossRef]
15. Mehdi, S.A.; Khalid, J.; Khayam, S.A. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In *Lecture Notes in Computer Science, Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, (RAID'11), Menlo Park, CA, USA, 20–21 September 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 161–180. [CrossRef]
16. Braga, R.; Mota, E.; Passito, A. Lightweight Ddos Flooding Attack Detection Using Nox/Openflow. In Proceedings of the IEEE Local Computer Network Conference, Denver, CO, USA, 10–14 October 2010; pp. 408–415. [CrossRef]
17. Kokila, R.T.; Selvi, S.T.; Govindarajan, K. DDoS Detection and Analysis in SDN-Based Environment Using Support Vector Machine Classifier. In Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC), Chennai, India, 17–19 December 2014; pp. 205–210. [CrossRef]
18. Phan, T.V.; van Toan, T.; van Tuyen, D.; Huong, T.T.; Thanh, N.H. OpenFlowSIA: An Optimized Protection Scheme for Software-Defined Networks from Flooding Attacks. In Proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha Long, Vietnam, 27–29 July 2016; pp. 13–18. [CrossRef]
19. Mousavi, S.M.; St-Hilaire, M. Early Detection of Ddos Attacks Against Sdn Controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, USA, 16–19 Febuary 2015; pp. 77–81.

20. Niyaz, Q.; Sun, W.; Javaid, A.Y. A deep learning based ddos detection system in software-defined networking (sdn). *arXiv* **2016**, arXiv:1611.07400. [CrossRef]
21. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. In Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016; pp. 258–263. [CrossRef]
22. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 202–206. [CrossRef]
23. Sen, S.; Gupta, K.D.; Manjurul Ahsan, M. Leveraging Machine Learning Approach to Setup Software-Defined Network(SDN) Controller Rules During DDoS Attack. In *Algorithms for Intelligent Systems, Proceedings of the International Joint Conference on Computational Intelligence, Dhaka, Bangladesh, 4 July 2019*; Uddin, M., Bansal, J., Eds.; Springer: Singapore, 2020.
24. Vetriselvi, V.; Shruti, P.S.; Abraham, S. Two-Level Intrusion Detection System in SDN Using Machine Learning. In *ICCCE 2018, Proceedings of the Lecture Notes in Electrical Engineering, Hyderabad, India, 24 January 2018*; Kumar, A., Mozar, S., Eds.; Springer: Singapore, 2019; Volume 500.
25. Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Machine-Learning Techniques for Detecting Attacks in SDN. *arXiv* **2019**, arXiv:1910.00817.
26. Dey, S.K.; Rahman, M.M. Flow based anomaly detection in software de-fined networking: A deep learning approach with feature selection method. In Proceedings of the 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 630–635. [CrossRef]
27. Dey, S.K.; Rahman, M.M.; Uddin, M.R. Detection of Flow Based Anomaly in Openflow Controller: Machine Learning Approach in Software Defined Networking. In Proceedings of the 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 416–421. [CrossRef]
28. Dey, S.K.; Uddin, M.R.; Rahman, M.M. Performance Analysis of SDN-Based Intrusion Detection Model with Feature Selection Approach. In *Algorithms for Intelligent Systems, Proceedings of the International Joint Conference on Computational Intelligence, Dhaka, Bangladesh, 4 July 2019*; Uddin, M.S., Bansal, J.C., Eds.; Springer: Singapore, 2020; pp. 483–494.
29. Louridas, P.; Ebert, C. Machine learning. *IEEE Softw.* **2016**, *33*, 110–115. [CrossRef]
30. Khan, G.M.; Khan, S.; Ullah, F. Short-Term Daily Peak Load Forecasting Using Fast Learning Neural Network. In Proceedings of the 2011 11th International Conference on Intelligent Systems Design and Applications, Cordoba, Spain, 22–24 November 2011; pp. 843–848. [CrossRef]
31. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the Kdd Cup 99 Data Set. In Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, (CISDA'09), Piscataway, NJ, USA, 8–10 July 2009; pp. 53–58. Available online: http://dl.acm.org/citation.cfm?id=1736481.1736489 (accessed on 6 March 2018).
32. Meng, Y. The practice on using machine learning for network anomaly intrusion detection. In Proceedings of the 2011 International Conference on Machine Learning and Cybernetics, Guilin, China, 10–13 July 2011; Volume 2, pp. 576–581. [CrossRef]
33. Yang, Y.; Pedersen, J.O. A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning, (ICML '97), Nashville, TN, USA, 8 July 1997; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA; pp. 412–420. Available online: http://dl.acm.org/citation.cfm?id=645526.657137 (accessed on 5 February 2019).
34. Ingre, B.; Yadav, A. Performance Analysis of Nsl-Kdd Dataset Using Ann. In Proceedings of the 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96. [CrossRef]
35. Breiman, L. *Machine Learning*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2001; pp. 5–32. Volume 45. [CrossRef]
36. Mandic, D.P.; Chambers, J. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2001.

37. Livieris, I.E. Forecasting Economy-Related Data Utilizing Weight-Constrained Recurrent Neural Networks. *Algorithms* **2019**, *12*, 85. [CrossRef]
38. Kolen, J.F.; Kremer, S.C. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*; IEEE: Piscataway, NJ, USA, 2001; Available online: https://ieeexplore.ieee.org/document/5264952 (accessed on 12 March 2018). [CrossRef]
39. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
40. Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using Rnn Encoder Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar; pp. 1724–1734. Available online: https://www.aclweb.org/anthology/D14-1179 (accessed on 22 April 2018). [CrossRef]
41. Chung, J.; Gu¨lc¸ehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR abs/1412.3555. *arXiv* **2014**, arXiv:1412.3555.
42. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
43. Nkiama, H.; Zainudeen, S.; Saidu, M. A subset feature elimination mechanism for intrusion detection system. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 148–157. [CrossRef]
44. Weka. Available online: https://www.cs.waikato.ac.nz/ml/weka (accessed on 22 March 2018).
45. Tensorflow. Available online: https://github.com/tensorflow (accessed on 30 March 2018).