Received December 20, 2021, accepted January 19, 2022, date of publication January 31, 2022, date of current version February 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3148134

Convolutional Neural Network Based Algorithm for Early Warning Proactive System Security in Software Defined Networks

AHMED H. JANABI[1,2], (Member, IEEE), TRIANTAFYLLOS KANAKIS[1], (Member, IEEE), AND MARK JOHNSON[1]
[1]Department of Computing, University of Northampton, Northampton NN1 5PH, U.K.
[2]IT Unit, Al-Mustaqbal University College, Babylon 51001, Iraq
Corresponding author: Ahmed H. Janabi (ahmed.janabi@northampton.ac.uk)

ABSTRACT Software-Defined Networking is an innovative architecture approach in the networking field. This technology allows networks to be centrally and intelligently managed by unified applications such as traffic classification and security management. Traditional networks' static nature has a minimal capacity to meet organisations business requirements. Software-Defined Networks (SDNs) are the emerging architectures that address a range of networking challenges with new solutions. Nevertheless, these centralised and programmable techniques face various challenges and issues that require contemporary security solutions such as Intrusion Detection Systems. Recently, the majority of this type of security solution has been developed using Machine Learning techniques. Deep Learning algorithms have recently been used to provide more accuracy and efficiency. This paper presents a new detection approach based on Convolutional Neural Network (CNN). The experiments proved that the proposed model could be successfully implemented in a Software-Defined Network controller to detect various attacks with 100% accuracy, achieved a low degradation rate of 2.3% throughput and 1.8% latency when executed in a large-scale network.

INDEX TERMS Deep learning-early warning proactive system (DL-EWPS), convolutional neural network (CNN), software-defined networking (SDN), intrusion detection system (IDS), deep learning (DL), RGB image, InSDN dataset.

I. INTRODUCTION

Security of modern communication networks alongside the integrity and privacy of data are growing to be a necessity in recent years. The essential requirement of an institution is to preserve its own and sensitive data from internal and external attackers. Some authorised users may infiltrate their company's data to be sent to others for various purposes. A continuous live feed of data causes difficulty in identifying an attack in real-time. Modern networks are composed of a combination of hardware and software entities tailored to meet the requirements of each organisation in a manner that appears arbitrary to the observer. These components include risks, weaknesses, and security limitations. The attack used by the software is complicated, and it puts the data at high risk. The developer and the programmers may ensure security in the systems by using log files. The complexity of modern communication networks makes systems vulnerable to security violations. The types of network attacks differ in severity and speedy based on different networking characteristics. The main challenge in modern network security is efficiently and in real-time detecting and mitigating the threats. Therefore, it is necessary to employ the Intrusion Detection System (IDS) technique to identify the internal and external intruders and protect the systems and networks.

Software-Defined Network (SDN) enables the network to be controlled by software that grans fewer networking devices and provides simple physical connectivity and configuration. Therefore, the network operators can adapt network behaviour to support modern services and security applications. Hence, the majority of the network's services will be more flexible, programmable, and not restricted by a platform [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrami.

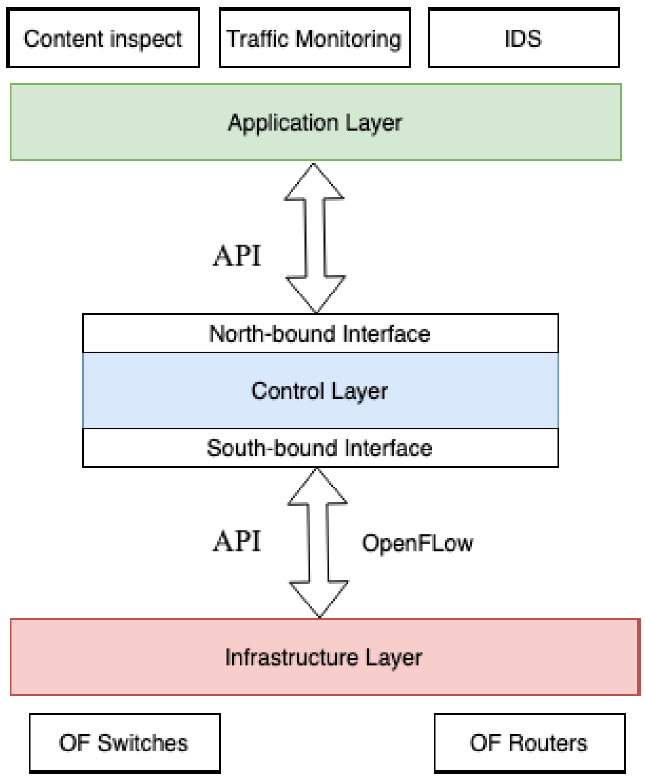


FIGURE 1. SDN structure [6].

The main SDN characteristic is that the control and data planes are separated using the Application Programmable Interface (API). This implies that SDN decouples the forwarding and control functions in the network [2]. The forwarding devices, such as switches and routers, are separated from the control logic and moved into the logical controller. The controller represents the centralisation in SDNs [2]. Therefore, the data plane consists only of forwarding device sets managed by the controller. The decoupling of the data and control planes allows the network's services and applications to be programmable to enhance scalability and reliability [2].

The layers and planes of the SDN architecture are shown in Fig. 1. As indicated, the administrator can activate the network's security and strategies in the application plane by redirecting the network traffic to several applications or systems by adopting the control plane [3]. OpenFlow represents the communication interface between the data and control planes. Therefore, the data plane devices require the OpenFlow protocol to be connected to the control plane [3].

SDN monitors the external and internal environments by using the software [4]. The main advantage of SDN in implementing an efficient IDS is active traffic observation, in which the controller can access network traffic when needed. Therefore, it has the potential to inspect any unusual traffic in a network [5]. An IDS can identify and locate malicious activities by monitoring the network's traffic offline or in real-time [6]. Existing IDSs in SDN are Signature-Based, and Anomaly-Based [6].

This paper presents new IDS developed for modern SDNs, named Deep Learning-Early Warning Proactive System (DL-EWPS). DL-EWPS is a new approach that can detect various attacks across the SDN network in real-time using the CNN classifier. CNN is one of the Deep Learning algorithms used in images classification, Analysing Documents, Face Recognition, etc [1]. The model detects various attacks efficiently, such as Denial of Service (DoS), Distributed Denial of Service (DDoS), Probe, User to Root (U2R), and Remote to local (R2L). The proposed model reduces the controller's overhead in existing IDSs. Therefore, the proposed model offers a lightweight algorithm for SDNs to confirm that the current request has been completed and then sends another request without stressing the controller. DL-EWPS also presents a new swift method to convert the numerical data to RGB images to be used by CNN classifier.

Additionally, we added new features extracted from the flow table statistics to increase the reliability of the proposed model. The added features have been selected based on the investigation and analysis of some typical attacks behaviours. These features are widely practical during attacks. The proposed system has been implemented in the SDN controller. The evaluation shows DL-EWPS can detect various attacks with 100% accuracy and achieve a low degradation rate of throughput and latency when executed in a large-scale network.

The rest of this paper is organised as follows: Section two presents a literature review, while Section three describes the proposed model and the advantages. Section four explains the feature selection and the dataset: Section five discusses the evaluation and the results. Furthermore, Section six included the conclusion of the proposed model and the future work.

II. LITERATURE REVIEW

An IDS has been shown in [7] called DDosTC utilising the hybrid neural network, which consists of flexible transformers and CNN algorithm. The proposed model achieved a prediction accuracy of 99.86% with a 6:4 (training: testing) ratio, which is considered relatively high. However, the used hybrid model increased the complexity of the model, which required more resources, such as memory and CPU processing. The proposed IDS can only detect DDoS attacks. In addition, the proposed model involves many extracted features that require extra memory and a long-time process. Therefore, it will cause the controller's bottleneck and cannot be implemented in real-time in large-scale networks. The majority of these extracted features are not related to the practices of DDoS attacks. A less complex model was presented in [8]. The authors proposed a Deep learning (DL) approach for the SDN context to identify the DDoS and DoS attacks between the controller and end-user devices in real-time. The proposed model in this research detects the attack by utilising the standard DL algorithm with Relu and Softmax functions. The CICIDS2017 public dataset was used for the training and testing phases. The proposed model achieved a prediction accuracy of 99.6%, which is also considered relatively high. However, the presented model also involves many extracted features that require high memory and a long CPU process. Some of these features are unreachable in the SDN context or require additional operations, leading to congestion on the OpenFlow channel and controller overhead.

Niyaz *et al.* [9] presented an IDS which is Deep Learning-based, identifying a DDoS attack in the SDN environment for multi-vector attack detection. The model includes three modules: Traffic Collector and Flow installer (TCFI), Feature Extractor (FE), and Traffic Classifier (TC). The system checks every packet in the SDN controller, extracts the features, and then passes the extracted features to be classified as either a regular or malignant packet. The authors collected a dataset from a home wireless network (HWN). In this paper, they employed the tcpdump tool and hping3 tools to generate DDoS traffic. At the Feature Extractor (FE) module, the proposed system extracts 68 features to be used in the classification for every single packet. However, it could only achieve a low accuracy of 95.65%. The number of extracted features requires high-level memory and a long processing time as this process performs for every packet. Therefore, this causes a bottleneck in the controller. In addition, most of these extracted features are not related to the practices of DDoS attacks. Moreover, the authors broke the concept of SDN architecture as they configured the controller to force the switches to send all packets through the controller to be treated and ignored the flow table function. Therefore, the controller will be crashed when running such a model in large-scale networks, resulting in a low performance in small networks.

On the other hand, in [10], a detection technique was proposed by employing a Deep Neural Network (DNN) approach to identify the DDoS attacks in the SDN networks. The NSL-KDD public dataset was used during the training and testing stages. The proposed model used the six basic features in the classification, and the controller extracts these features for each flow in real-time. However, the small number of the extracted features caused too low accuracy in the detection stage, which was 75.75%. These features have been extracted from the flow's basic statistics information and are not sufficient to cover the attacks' behaviours; therefore, the attacker can easily avoid the IDS. The authors configured the controller to receive flow statistics from all OpenFlow switches sequentially each fixed time. This process stresses the controller, but not more than in the model proposed in [9].

A comprehensive IDS is presented in [11]. The proposed model is a flow-based anomaly detection approach in the OpenFlow controller that uses Gated Recurrent Unit Long Short-Term Memory (GRU-LSTM DNN). A feature selection technique called ANOVA F-TEST was used to obtain a high-performance classification. The NSL-KDD public dataset was utilised for the training and testing stages of the experiment. The proposed model detects various attacks such as Prop, U2R, R2L, and DoS. The proposed model employed 41 features in the classification extracted for each flow at the SDN controller in real-time. However, the model achieved an accuracy of 87% with a false alarm rate of 0.76% which are considered relatively low. The proposed system is complex as more DL algorithms have been used in feature selection and classification. Such algorithms utilise the resources. In addition, the presented model also involves a large number of extracted features that require high memory and a long CPU processing time. Hence, the controller will generate an overload, negatively affecting the network's performance. In contrast, a lightweight method was proposed in [12] to reduce the controller overload. This approach utilises the Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) to classify traffic in real-time. The NSL-KDD public dataset was used during the training and testing phase. The proposed model used six basic features for the classification. These features are extracted for each flow inside the SDN controller in real-time. However, the presented model achieved a low throughput degradation rate of 7% and achieved a low detection rate of 89%. This limited number of extracted features has resulted in low efficiency in attack detection because it is primitive and does not sufficiently cover the attack behaviours, especially when traffic is real-time and threats can harm the communication in a short time.

Jiaqi *et al.* [13] proposed an ML approach in the SDN 5G environment for identifying DDoS, DoS, U2R, and R2L attacks. The proposed system is executed using the SDN controller. The K-means++ and AdaBoost algorithms were used to classify the traffic, and the Random Forest (RF) algorithm was used for feature selection. The authors used the KDD Cup 1999 dataset, which is widely used to evaluate the IDSs. However, the proposed model in this paper uses a Big Data Center module to store the network packets for further analysis, resulting in increased model complexity. The model has been achieved an accuracy of 84%, which is considered relatively low. The Random Forest (RF) algorithm failed to select related features that cover the behaviours of the attack, and this can be observed when the model achieves a low detection accuracy. The authors did not consider the controller's bottleneck as the controller requests all flows from flow tables while it has not yet finished the previous request. Therefore, the presented model requires a technique to confirm that the current process has been completed and then sends other flow statistics requests. Conversely, Hybrid Neural Network (HYBRID-CNN) method was used in [14] in order to detect the attacks. The model has been achieved an accuracy of 95.64% which is considered relatively high. The proposed method is highly complex because it combines two Deep Learning (DL) algorithms: Deep Neural Network and Convolutional Neural Network. The nature of these algorithms is high resources utilisation. Therefore, the proposed system is inappropriate with a large network.

Alternatively, in [15], an approach to identify attacks was presented. The neural network algorithm was used to classify each flow inside the SDN controller. The NSL-KDD public dataset is used to train and test the proposed IDS.

The model's target is to define the DOS, U2R, R2L, and Probes. The model achieved a detection rate of 97.4%. However, in the training stage, the use of a small dataset size negatively affects the actual test's detection accuracy. The extracted features are obtained only from the header packet, which are not cover the attack behaviours. In addition, the dataset used has a highly redundant record. The processing operation was conducted in the controller for each packet involved in generating a bottleneck for the controller and a single point of failure. This is a big challenge in implementing Machine Learning (ML) approaches with IDS. Additionally, the proposed model requires a feature selection method to select the features that are effective when an attack occurs.

The essential issues with ML/DL approaches are extraction and feature selection methods. Existing systems suffer from many features used in classification or use only basic features. Using a large number of features causes overload and problematic latency in the network. While using a small number/basic features does not provide accurate detection of attacks as it does not cover the behaviour of attacks. Therefore, obtaining a high accuracy with limited raw features using DL/ML approaches requires more attention from researchers.

Most researchers propose models that require monitoring traffic in real-time; therefore, this process needs to analyse each packet/flow pass through the network and then classify it as normal or malignant by using a specific classifier. Therefore, the processing requires time, memory, and more CPU usage to collect, process, and classify the traffic. This process causes congestion in the controller and the switches. Therefore, researchers must consider the trade-off between classifiers efficiency and network performance.

III. SYSTEM MODEL

DL-EWPS is a flow-based IDS for the SDN networks, as shown in Fig. 2. The system consists of three modules: Flow Table Statistics' Sender, Statistics' Receiver and Features' Extractor, and Flow Classification and Counter. The model was designed to identify DoS, DDoS, Port Scan, U2R, and R2L attacks in real-time. The proposed system exhibits the following properties:

1) Flexibility: It is easy to add more types of attacks by feeding the model new data, including new threats. The system can be configured and operated using an OpenFlow device.
2) Scalability: The proposed system can be used for large-scale and small networks. In addition, it is fixable to be implemented in networks that have more than one controller.
3) Swiftly: The proposed model offers swift methods to perform the IDS and significantly decreases the controller overhead.
4) Reliability: DL-EWPS is reliable as an accurate dataset, and efficient algorithms have fed it.

A. FLOW TABLE STATISTICS' SENDER MODULE

This module collects the statistics of each flow for all switches. After the OpenFlow switch receives the ofp_flow_stats request message from the controller, the switch responds and sends the flow table details. The switch sends those information to the controller via an ofp_flow_stats reply message through the OpenFlow channel [16]. In this module, we added a new indicator that refers to the last update time for each flow. This indicator is used by the Statistics' Receiver and Features' Extractor module to extract the features for each received flow that is updated within the last 3 seconds.

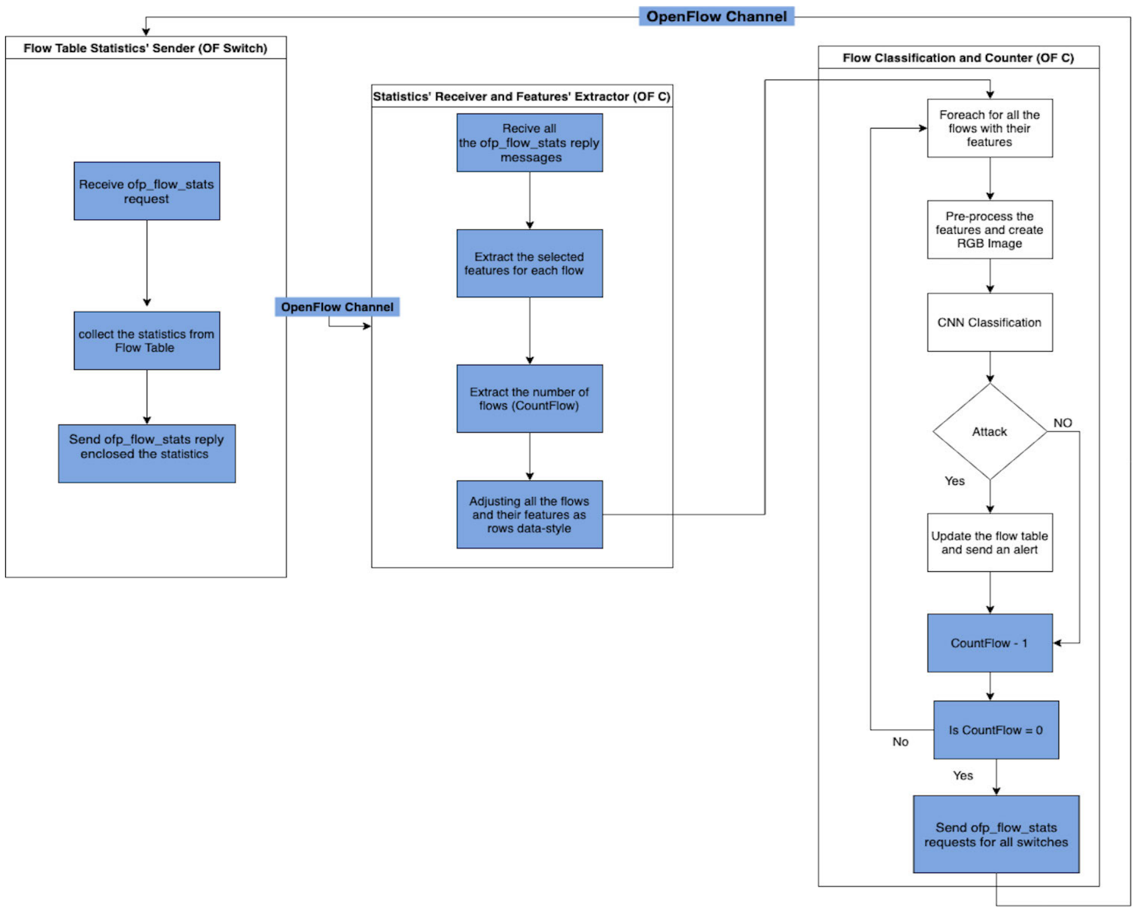
B. STATISTICS' RECEIVER AND FEATURES' EXTRACTOR MODULE

This module receives all flow table statistics of each connected OpenFlow switch after being sent by the Flow Table Statistics' Sender module, as shown in Fig. 2. This model is the first step process that the controller must perform. After receiving all ofp_flow_stats reply messages, the module extracts the selected eleven features in Table 2 for each updated flow in the last three seconds. Three seconds interval time was selected to obtain sufficient information to identify the attack [17]. The too-long interval time allows the controller and switches to process many flows, which negatively affects the network's performance and sometimes causes controller crashes. The module then calculates the number of received messages and the number of flows (CountFlow) to be used in the next module.

C. FLOW CLASSIFICATION AND COUNTER MODULE

This module is being enabled when the features have been extracted. After the essential process is complete, the module will receive all the currently requested flows from the Statistics' Receiver and Features' Extractor module. At this time, each flow was enclosed by its features and statistics. Afterwards, the module performs a pre-processing step for each flow. In the pre-processing stage, the module converts the numerical features into RGB images to be classified utilising the CNN algorithm.

The purpose of converting numerical data into RGB images is that the CNN classifier is designed for images classification. Therefore, the CNN requires images as input to work appropriately [18]. Hence, the proposed model presents a novel method to generate RGB images using the selected eleven features, as demonstrated in Fig. 3. These features are numerical; to convert it to an RGB image, the model demands to re-scale the data within the range of 0-255 because the RGB image takes values within that range for red, green, and blue. The RGB image consists of a set of pixels, and each pixel has three bytes acting on the R, G, and B values [19]. To adapt that with the given numerical data, we selected a 3×4 array to cover all the selected eleven features. Each location in the array has three identical values. Only the last location in the array takes three different values, where R is the value of the first feature, G is the value of the fifth feature, and B is



OpenFlow Channel
Flow Table Statistics' Sender (OF Switch)
Receive ofp_flow_stats request
collect the statistics from Flow Table
Send ofp_flow_stats reply enclosed the statistics
OpenFlow Channel
Statistics' Receiver and Features' Extractor (OF C)
Recive all the ofp_flow_stats reply messages
Extract the selected features for each flow
Extract the number of flows (CountFlow)
Adjusting all the flows and their features as rows data-style
Flow Classification and Counter (OF C)
Foreach for all the flows with their features
Pre-process the features and create RGB Image
CNN Classification
Attack
NO
Yes
Update the flow table and send an alert
CountFlow - 1
Is CountFlow = 0
No
Yes
Send ofp_flow_stats requests for all switches

FIGURE 2. The system architecture.

the value of the eleventh feature. These specific features were selected to take the location (3,2) of the image array because these features are widely affected when an attack occurs. For example, the first feature's value is 33.36, after re-scale it by equation 1, it will be equalled to 0.2, so the R, G, and B values will be equal to (0.2, 0.2, 0.2), and these values will be settled in location (0,0) of the array. The combination of these values gives a special colour, as depicted in Fig. 2.

$$New\_Value = \frac{x - min}{max - min} \times 255 \tag{1}$$

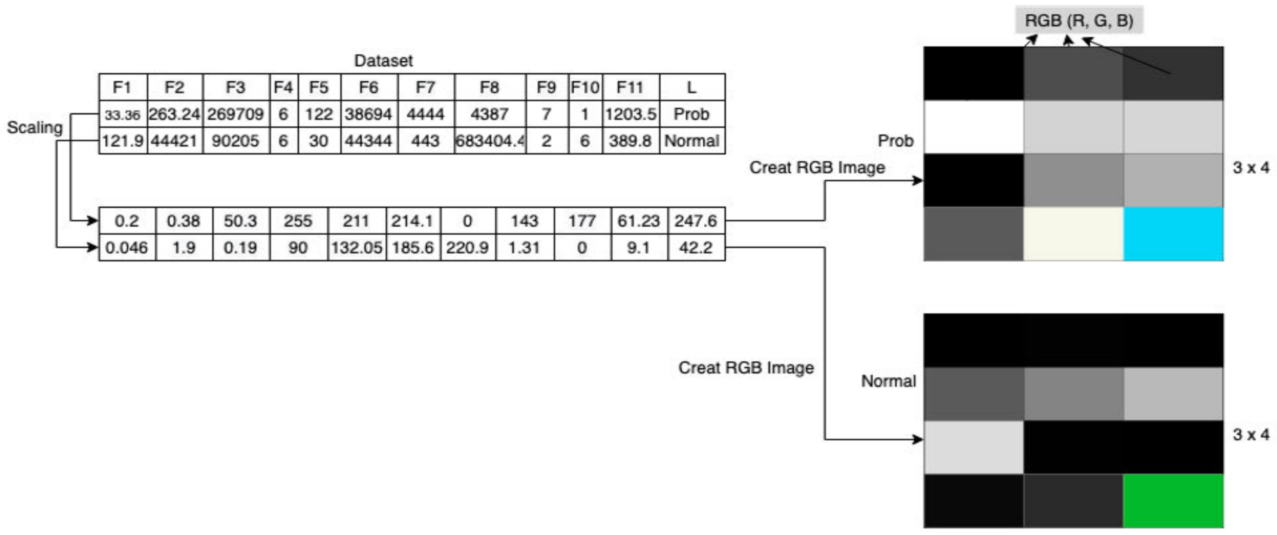
where x is an original value, New_Value is the re-scaled value.

The converted RGB image will be forwarded to the pre-trained CNN classifier. The CNN classifier will predict forwarded image according to its analysing. When CNN has performed the forecasting process, the result will be allocated to the type of the current flow, whether normal or attack. The CNN model was fully trained by using the InSDN dataset [20]. In the experiments, we employed Keras and TensorFlow to develop the CNN model [21], [22]. The specifications of the CNN model are listed in Table 1. These specifications have been selected through several experiments. It was noted that the use of Epoch 10 with a batch size of 32 is more convenient and contributes to reducing the training time. Fig. 4 shows the architecture of the proposed CNN algorithm. The CNN architecture included two Convolutional 2D and two Max Pooling Layers.

TABLE 1. CNN model specifications.

Variable	Parameters
Convolutional2D	2 Layers
MaxPooling	2 Layers
Flatten	5000
Ouputs	2
Activation Function	Relu
Optimizer Function	Adam
Loss Function	Sparse Categorical Cross Entropy
Epoch	10
Kernel Size	3 * 3
batch_size	32

If the current flow is identified as normal, the counter flow (CountFlow) will be decreased by one, and then will take the following flow within the current request. The controller sends two orders if the current flow is identified as attack flow. The first action will send to the switch to update the flow table of the predicted flow by sending of.OFPP_DROP request. This order orders the switch to drop any further



F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	L
33.36	263.24	269709	6	122	38694	4444	4387	7	1	1203.5	Prob
121.9	44421	90205	6	30	44344	443	683404.4	2	6	389.8	Normal

0.2	0.38	50.3	255	211	214.1	0	143	177	61.23	247.6
0.046	1.9	0.19	90	132.05	185.6	220.9	1.31	0	9.1	42.2

FIGURE 3. Convert the features to RGB image.

matching packets in the future. The second action will be forwarded to the administrator to take further actions to minimise the risk of the detected malignant traffic. In this case, the counter of the flows (CountFlow) will also be decreased, and the module then processes the following flow in the current request to be processed using the same procedure that followed previously. If the CountFlow counter has elapsed, the model will send new ofp_flow_stats request messages for all the connected switches, as shown in Fig. 1 and Algorithm 1. This process is novel and reduces the controller's overload and OpenFlow channel congestion by confirming that the current request has been completed and then sending another request without stress to the controller and the OpenFlow channel.

IV. FEATURES SELECTION AND DATASET

Some of the researchers have used only basic flow statistics extracted directly from the flow tables. Such basic

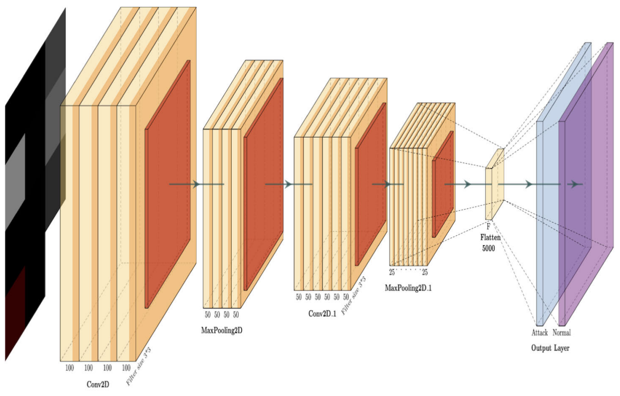


FIGURE 4. CNN architecture.

Algorithm 1: Counter Algorithm

```
Data:
$FlowCount \leftarrow Length(ofp\_flow\_stats[S1]) + \ldots +$
  $Length(ofp\_flow\_stats[Sn]));$
$All\_Flows \leftarrow all\_ofp\_flow\_stats\_replies$
for $flow$ in $All\_Flows$ do
    $NewFlow \leftarrow Rescale(flow);$
    $FlowImage \leftarrow ToRGB(NewFlow);$
    $Predict \leftarrow CNN\_Model(FlowImage);$
    if $Predict$ is $Attack$ then
        $Send(of.OFPP\_DROP);$
        $Send(alertToAdmin);$
        $FlowCount \leftarrow FlowCount - 1;$
    else
        $FlowCount \leftarrow FlowCount - 1;$
    end
    if $CountFlow == 0$ then
        $Send(ofp\_flow\_stats\_request)$
    else
        Next
    end
end
```

statistics: duration, protocol_type, count_bytes, src_port, and count_packet [23], [24]. These statistics are inadequate for identifying attacks efficiently and do not cover the attack's behaviours [24]. This issue negatively reflects the reliability of the IDSs to detect the attacks. In addition, the high number of extracted features also affects the network's performance, as it requires more processing time and consumes the network resources.

To solve the above issues, we fed the model with eleven features to increase the reliability of the proposed model, which covered most of the attacks behaviours, as shown in

Table 2. These features were chosen depending on how its effects when an attack occurred. This feature number is not large; hence, it avoids the long time and complexity of the process.

We used the InSDN dataset to train our DL-EWPS. The InSDN is a new dataset created by the University College Dublin team in September 2020 [20]. This dataset was generated by a mechanism that is realistic. Most of the public dataset issues are solved in the InSDN, such as redundancy [20]. The InSDN dataset is the first public dataset created within the SDN architecture. Furthermore, this dataset includes updated attacks and simulates new behaviours. The InSDN includes the attacks such as DoS, DDoS, Prob, U2R, R2L, and Portscan. The InSDN dataset is the first public dataset collected from the SDN environment [20].

In addition, we added new features extracted from the flow table statistics to increase the reliability of the proposed model. These features are the following, Con_s_service, ave_bytes, N_uiqIP, N_reDstIP, and Pkt_Avg, as described in Table 2. The added features were selected based on the investigation and analysis of some typical attack behaviours. These features have been evaluated and clearly observed that are effective when an attack occurs, as shown in Fig. 5. Therefore, DL-EWPS has another contribution when these features are added. Our experiments showed that the model reduced the overhead on the controller and increased the accuracy of the CNN classifier, as described in the next section.

TABLE 2. The selected features description.

Feature	Description
Flow Pkts/s	Number of flow packets
Flow Byts/s	Number of flow bytes
Flow Duration	Duration of the flow in seconds
Protocol_type	Protocol type
Con_s_service	Number of flows that have same service connection
Src Port	Port number of the source
Dst Port	Port number of the destination
ave_bytes	Average of bytes in flows that have the same connection
N_uiqIP	Number of the unique source IP addresses
N_reDstIP	Number of the repeated destination IP
Pkt_Avg	Average size of packet

V. SYSTEM MODEL EVALUATIONS AND EXPERIMENTS

A. THE DETECTION TECHNIQUE EVALUATION

We evaluated the DL-EWPS through various experiments. We used the following metrics to test and evaluate the effectiveness and performance of the detection technique in our model.

- False Positive (FP): The number of regular traffic is incorrectly classified as attack traffic.
- False Negative (FN): The number of attack traffic is wrongly classified as regular traffic.
- True Positive (TP): The number of attack traffic classified correctly as attack traffic.
- True Negative (TN): The number of normal traffic that is classified correctly as regular traffic.

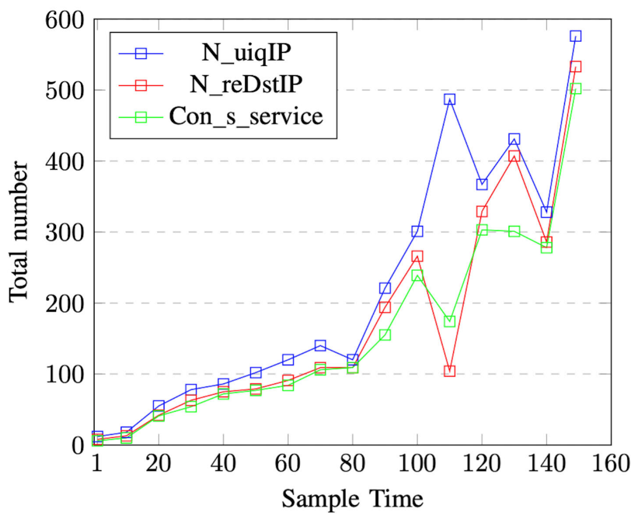


FIGURE 5. The impact of the added features at attack.

- Accuracy (AC): Percentage of accurate predictions over the total traffic. The following equation has been used to calculate the accuracy:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{2}$$

- Precision (P): also known as a false alarm, it is used to assess the number of attack traffic detected correctly. P calculates by equation 3:

$$P = \frac{TP}{TP + FP} \times 100\% \tag{3}$$

- Recall (R): estimates the rate of predicted attacks against the total of attack traffic. R is calculated by follows equation:

$$R = \frac{TP}{TP + FN} \times 100\% \tag{4}$$

- F1-measure (F1): returns a more reliable measurement of the model's accuracy depending on both R and P.

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \times 100\% \tag{5}$$

The CNN model was evaluated by using the above metrics with a binary labelled dataset (Normal and Attack). The CNN algorithm has been selected to classify the sample flows in real-time. The architecture and CNN specifications are shown in Fig. 4 and Table 1. The CNN was implemented by using the TensorFlow and Keras libraries in Python. TensorFlow and Keras libraries developed specifically for ML and DL implementation in Python language [21], [22]. The experiments were conducted by training and testing the model using three feature groups of the InSDN dataset. The feature groups are as follows,

- Basic Features Group: includes only basic statistics directly extracted from the flow table. These features include Flow Pkts/s, Flow Byts/s, Flow Duration, Protocol_type, Src Port, and Dst Port. The researchers have widely used these Features Group. These features sets are rudimentary and do not cover the behaviours of attack [24].

- All features Group: includes all features sets in the InSDN dataset, which includes 83 features extracted for each flow [20].
- 11 Features Group: includes the basic features and five new features added to cover the behaviour of attacks and increase the model efficiency. These features are listed and described in Section IV and Table 2.

TABLE 3. Metrics evaluation with different features groups.

Features Group	AC	P	R	F1
Basic features	96.43%	96%	95.6%	95.7%
All features	98.94%	98%	98.4%	98.2%
11 features	**100%**	**100%**	**100%**	**100%**

The results of these experiments are shown in Table 3. Our CNN achieved the highest accuracy, P, R, and F1. In all features groups, the proposed model has achieved the highest accuracy of 100%. This proves that the proposed model is promising and can detect attacks with a 100% accuracy. The CNN model was evaluated using a 5-fold cross-validation method in all scenarios.

TABLE 4. Accuracy comparison with other classifiers.

Classifier	Basic features	All features	11 features
NB	88.957%	97.367%	98.527%
DNN	75.50%	84.77%	85.32%
Logistic Regression	89.61%	93.11%	98.42%
SVM	76%	71%	78%
Our CNN	**96.43%**	**98.94%**	**100%**

We compared our model with other classifiers, such as NB, DNN, SVM, and Logistic Regression. The proposed system achieved the highest accuracy among the other classifiers, as shown in Table 4.

We can conclude from the experiments that the CNN architecture and the invented method of generating RGB images contributed to achieving highly accurate attack detection.

B. NETWORK PERFORMANCE EVALUATION

It is vital to evaluate the network performance when installing an IDS to check the integration. The most commonly used metrics in network performance evaluation are throughput and latency. Therefore, the throughput and latency metrics have been utilised to evaluate our DL-EWPS.

1) THROUGHPUT EVALUATION

In general, throughput is defined as the number of elements passing through a process. In this step, DL-EWPS was tested using the Cbench tool. Cbench is a benchmark equipment employed to evaluate the throughput and latency of the controller [17]. This tool is open-source and programmed in C language and is available online on [25]. Cbench calculates how many packets the controller can treat per second. In throughput mode, Cbench sends extensive packet_in requests to the controller by all switches and calculates how many packet_out replies have been returned per second.

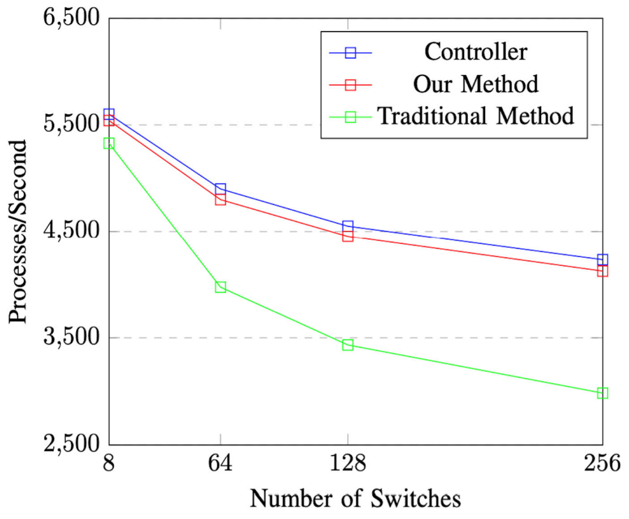


FIGURE 6. Throughput evaluation.

We used the Mininet emulator and POX controller to implement the DL-EWPS and test the network performance with the throughput evaluation. The implementation was performed utilising six scenarios. Each scenario used a specific number of OpenFlow switches: 8, 16, 32, 64, 128, and 256. These switches were connected to a single controller via port number 6633. Each switch had 1000 end-users in all scenarios. All scenarios were tested using our features extraction method and the traditional method. Our features extraction was performed in two phases. The first phase extracts only the features of flows that are updated within the last 3 seconds to minimise the generated overhead on the controller. In the second phase, the controller does not send further ofp_flow_stats requests to the switches without completing the processing of the current request. This phase reduced the overload on both the controller and OpenFlow channel. In comparison, many researchers have configured the controller to send requests during a fixed period, regardless of the current request has been processed or not. Hence, such a process will lead to exhausting the OpenFlow channel and the controller.

As shown in Fig. 6, the DL-EWPS does not have a high controller's degradation rate. The highest degradation rate was 2.3% when the controller was tested using 256 switches. The degradation rates of the other scenarios were between 1% and 2%. These rates are insufficient and do not affect the controller performance when the DL-EWPS is installed in the POX controller. Therefore, the throughput evaluation proves that the proposed DL-EWPS is promising and highly efficient in large-scale networks.

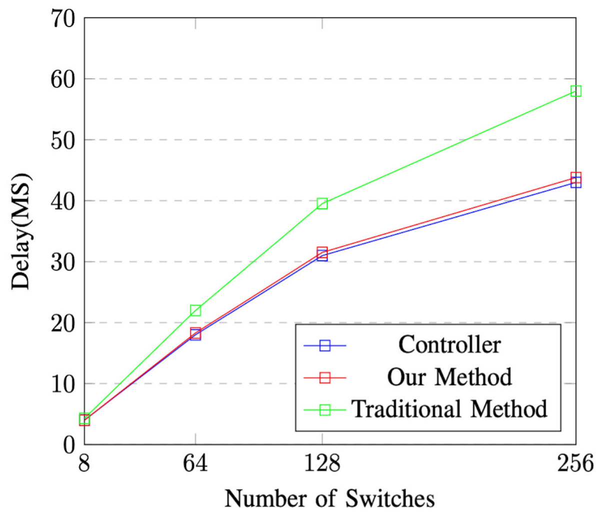


FIGURE 7. Latency evaluation.

In contrast, as shown in Fig. 6, we tested the topology using the traditional method to coach all flows in the OpenFlow tables of the connected switches. The controller sorts the data and extracts the features for all flows. This process increased the overhead/congestion of the controller and decreased the throughput to a degradation rate of approximately 29.5% when 256 switches were used.

2) LATENCY EVALUATION

Latency is defined as the time required for packet_in to reach a target [26]. In the latency evaluation, we used a series of the same experiments used in the throughput evaluation. In latency mode, Cbench makes all connected switches send a single packet_in request to the controller and calculates how long it will take to hit the destination [17].

As shown in Fig. 7, DL-EWPS does not cause a high latency rate. The highest latency rate was 1.8% when the controller was tested using 256 switches. Moreover, the latency rates of the other scenarios were between 1.4% and 1.8%. These rates do not affect the network performance when installing the DL-EWPS. Accordingly, the latency evaluation also proves that the proposed model is highly efficient for large-scale networks.

As presented in Fig. 7, the topology was tested utilising the traditional method to coach all the flows in the OpenFlow tables of the connected switches. This process increased the latency rate to 34.8% when 256 switches were used. Therefore, this rate is considered very high and causes deficient network performance.

VI. CONCLUSION

This paper presented a new DL-EWPS model that can predict the network attacks early using a Deep Learning approach in SDN networks. The model employed a novel method to convert the numerical data into RGB images to benefit from the advantages of a CNN-based classifier. This method increased the accuracy of the CNN predictor and reduced the size of the images. In addition, we added new features extracted from the flow table statistics to increase the reliability of the proposed model. The added features were selected based on the investigation and analysis of some typical attack behaviours.

Through the experiments and evaluations, the DL-EWPS achieved an accuracy of 100% in traffic classification. At the same time, the latency and throughput evaluations prove that our model is promising and has high efficiency for large-scale networks. In general, our proposed system is the first model in SDN to achieve 100% accuracy with significant performance when installed on large-scale SDN networks.

In future work, we intend to implement our proposed detection system on physical devices, where all the researchers use emulation to evaluate their systems.

REFERENCES

[1] J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, p. 916, Jun. 2020.

[2] R. Palanikumar and K. Ramasamy, "Software defined network based self-diagnosing faulty node detection scheme for surveillance applications," *Comput. Commun.*, vol. 152, pp. 333–337, Feb. 2020.

[3] Y. Goto, B. Ng, W. K. G. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic OpenFlow–based switch model," *Comput. Netw.*, vol. 164, Dec. 2019, Art. no. 106892.

[4] A. Shaghaghi, M. A. Kaafar, R. Buyya, and S. Jha, "Software-defined network (SDN) data plane security: Issues, solutions, and future directions," in *Handbook of Computer Networks and Cyber Security*, B. Gupta, G. Perez, D. Agrawal, and D. Gupta, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-22277-2_14.

[5] K. Bhushan and B. B. Gupta, "Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 5, pp. 1985–1997, May 2019.

[6] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[7] H. Wang and W. Li, "DDosTC: A transformer-based network attack detection hybrid mechanism in SDN," *Sensors*, vol. 21, no. 15, p. 5047, Jul. 2021.

[8] S. Boukria and M. Guerroumi, "Intrusion detection system for SDN network using deep learning approach," in *Proc. Int. Conf. Theor. Applicative Aspects Comput. Sci. (ICTAACS)*, Dec. 2019, pp. 1–6.

[9] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," *EAI Endorsed Trans. Secur. Saf.*, vol. 4, no. 12, p. e2, 2016.

[10] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263.

[11] S. K. Dey and M. M. Rahman, "Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method," in *Proc. 4th Int. Conf. Electr. Eng. Inf. Commun. Technol. (iCEEiCT)*, Sep. 2018, pp. 630–635.

[12] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 202–206.

[13] J. Li, Z. Zhao, and R. Li, "Machine learning-based IDS for software-defined 5G network," *IET Netw.*, vol. 7, no. 2, pp. 53–60, Mar. 2017.

[14] P. Ding, J. Li, L. Wang, M. Wen, and Y. Guan, "HYBRID-CNN: An efficient scheme for abnormal flow detection in the SDN-based smart grid," *Secur. Commun. Netw.*, vol. 2020, pp. 1–20, Aug. 2020.

[15] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Proc. 7th Int. Conf. Emerg. Secur. Technol. (EST)*, Sep. 2017, pp. 138–143.

[16] P.-W. Chi, C.-T. Kuo, J.-W. Guo, and C.-L. Lei, "How to detect a compromised SDN switch," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, Apr. 2015, pp. 1–6.

[17] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, "DeepIDS: Deep learning approach for intrusion detection in software defined networking," *Electronics*, vol. 9, no. 9, p. 1533, Sep. 2020.

[18] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, M. Adam, A. Gertych, and R. S. Tan, "A deep convolutional neural network model to classify heartbeats," *Comput. Biol. Med.*, vol. 89, pp. 389–396, Oct. 2017.

[19] Y. Yan, L. Zhang, J. Li, W. Wei, and Y. Zhang, "Accurate spectral super-resolution from single RGB image using multi-scale CNN," in *Proc. Springer Chin. Conf. Pattern Recognit. Comput. Vis. (PRCV)*, 2018, pp. 206–217.

[20] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020.

[21] TensorFlow. *TensorFlow 2 Quickstart for Experts*. Accessed: Aug. 20, 2021. [Online]. Available: https://www.tensorflow.org

[22] Keras. *Developer Guides*. Accessed: Aug. 20, 2021. [Online]. Available: https://keras.io

[23] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in SDN," in *Proc. IEEE 7th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Dalian, China, Oct. 2019, pp. 277–281.

[24] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM workshop Hot topics Softw. defined Netw. - HotSDN*, 2013.

[25] (2013). Cbench. *Github*. Accessed: Jan. 26, 2021. [Online]. Available: https://github.com/mininet/oflops/tree/master/cbench

[26] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, "Minimum control latency of SDN controller placement," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 2175–2180.

AHMED H. JANABI (Member, IEEE) received the Bachelor of Engineering degree (Hons.) in computer networking, in 2017.

Previously, he studied information technology at the University of Babylon, Iraq. He is currently a Researcher with the Faculty of Arts, Science, and Technology, University of Northampton, U.K. His previous positions include a Teaching Assistant within the University of Northampton, the University of Babylon, and the Al-Mustaqbal University College. He is in the progress of a journal write-up in the field of software-defined networks and security. His research interests include (but not limited to) software defined networks, the Internet of Things, cyber security on network level, computer architecture and engineering, and microprocessors.

TRIANTAFYLLOS KANAKIS (Member, IEEE) received the B.Eng. degree from the University of Northampton (UoN), U.K., the M.Sc. degree from the King's College of London, the M.B.A. degree from UoN, and the Ph.D. degree from the University of Greenwich.

He is currently the Program Leader of computer networks engineering at UoN. He has more than 15 years of experience in cellular and wireless communications engineering, multiple antenna systems, and cooperative communications. He has numerous publications in leading journals and conferences, while he is the coauthor to the Best Paper Award winning work in forward error correction presented in IEEE CEEC 2016. He is the Main Organizer of UoN IoT Workshop and an Academic Advisor of the IEEE Student Branch Manager at UoN. He has long experience in the industry, where he has worked as a Network Specialist at AT&T, a Technical Consultant and a Technical Trainer in the LTE4G era, and more recently, a Technology Consultant in the constructions sector while he has served the Hellenic Navy as part of his national military service. He has been the keynote speaker in numerous events worldwide. He is a fellow of the Higher Education Academy (FHEA). He has acted as a technical committee member of various conferences in the field of telecommunications engineering while he has served as a Reviewer for numerous conferences and journals, including IEEE Access.

MARK JOHNSON received the Ph.D. degree from the University of Northampton (UoN), in 2007. He is currently the Program Leader of the software engineering course at UoN. He is also the Deputy Subject Leader for technology. He is also a Key Consultant at Nyx Technologies Ltd., and works on a variety of contemporary software and hardware-based projects. He has more 20 than years of experience as a Software Engineer and an Academician. He has a significant research interest in pedagogic research within computing and has worked with several commercial organizations, including Barclaycard, Triad Group PLC, and KPMG, exploring the viability of student engagement with industry as an essential component of undergraduate course provision. As an Academician, he has supported the development of over 20 computing course programs (at undergraduate and postgraduate level) during his 22 years in higher education. He is a fellow of the Higher Education Academy.