symmetry

MDPI

Article

ShChain_3D-ResNet: Sharding Blockchain with 3D-Residual Network (3D-ResNet) Deep Learning Model for Classifying DDoS Attack in Software Defined Network

E. Fenil [1] and P. Mohan Kumar [2,*]

[1] Department of Computer Science and Engineering, Ponjesly College of Engineering, Nagercoil 629003, India; fenildanlinsam@gmail.com
[2] Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore 641008, India
* Correspondence: mohankumarmohan@gmail.com

Abstract: The distributed denial of service (DDoS) vulnerabilities have rapidly extended and have been given different possibilities for even more advanced assaults on specific targets in recent times, thanks to the growth of innovative technology such as the Internet of Things (IoT) and Software-Defined Networking (SDN). The attack patterns route comprises unprotected and susceptible IoT systems that are internet-connected, as well as denial of service weaknesses in the SDN controllers, such as southbound connection exhaustion. (1) Background: The review does not go into detail about the symmetry blockchain approaches used to mitigate DDoS attacks, nor does it classify them in IoT; (2) To overcome the privacy issues, a novel deep learning-based privacy preservation method was proposed named ShChain_3D-ResNet. This novel method combines Sharding, blockchain and Residual Network for securing the SDN. Under this network, the proposed efficient attention module jointly learns attention to enforce the symmetry on weights for various channels in spatial dimension as well as attention weights of multiple frames in temporal dimension assistance of pre-training, updating, and dense convolution process; (3) Results: the proposed ShChain_3D-ResNet achieves 95.6% of accuracy, 97.3% of precision, 95.2% of recall, 94.4% of F1-score, 32.5 ms of encryption time and 35.2 ms of decryption time for dataset-1. Further, it achieves 97.3% accuracy, 95.3% precision, 96.1% recall, 98.2% F1-score, 32.1 ms of encryption time, and 36.2 ms of decryption time for dataset-2; (4) Conclusions: The Sharding strategy can increase ShChain performance while simultaneously utilizing Multi User (MU) resources for SDN.

Keywords: deep learning; privacy; blockchain; Software Defined Network (SDN); neural network security

Citation: Fenil, E.; Mohan Kumar, P. ShChain_3D-ResNet: Sharding Blockchain with 3D-Residual Network (3D-ResNet) Deep Learning Model for Classifying DDoS Attack in Software Defined Network. *Symmetry* **2022**, *14*, 1254. https://doi.org/10.3390/sym14061254

Academic Editors: Chin-Ling Chen, Jeng-Shyang Pan and Sergei D. Odintsov

Received: 10 May 2022
Accepted: 13 June 2022
Published: 16 June 2022

1. Introduction

SDN liberates network topology from the constraints of traditional network intricacy and dependency, allowing it to meet adaptability, dependability, and privacy at the same instance. It distinguishes the control layer and data layer, as well as the platform's control and data relaying functions [1]. The responsibility of the control layer is taking routing decisions, with the support of just transmitting traffic and doing other tasks. The distance between the two planes improves the network's abstraction and development capabilities while also making the network model less tiresome and wasteful. The interconnectedness issue is fixed in three dimensions: network control, data plane, and application plane via central control and realization of new programming. The innovative API links the network control with the application plane. The Protocol stack is the fundamental component of the downstream API, which allows the central controller to interact with the data layer. SDN basically acts as a central nervous system (manager) for the network elements, allowing administrators to remotely program each networking layer component [2] by applying symmetry in each layer and have a total layer with the usage of software to govern

network activities via a centralised command post. Virtualization technology provides several benefits over previous network designs, however it is frequently exposed to system dangers and assaults. Security device licensing and global view collection are the most commonly used network risks against SDN [3,4]. Physical security features do not even have the choice to determine since packets are delivered as per flow control, and hackers can overcome security features until they are deployed. The operator is at the heart of the infrastructure and has access to a variety of network information data. The intruder may utilise the controllers to gain direct access to the network's wide perspective and execute significant assaults. SDN seems to have a distinct planar layout, with multiple attack objects at various planes. Since the manager oversees the whole network at the control and data planes, an intruder harming the controllers can prove to be detrimental. In the latest days, managers have been the primary focus of assaults. The following are the major security problems on the control and data planes:

Control plane service attacks: They may pose a major danger to the control plane's integrity.

Scalability Attacks: Absence of adaptability causes the management plane to become overcrowded and unable to manage additional streams [5,6].

DDoS attack: DDoS attacks were susceptible to operators [7]. Whenever a packet is received, the toggle compares it to the route table record. The toggle encapsulates the incoming packets in such a protect the contents statement and transmits the packet to the controller, in case of packet failure. The switch passes a significant number of data to the control layer, and at once the attacker delivers a high number of packets to the network. The device's resources become scarce by a significant number of DDoS spoofing packets, rendering itself unable to function correctly, creating severe risk to the device as well as the core system.

In a DDoS attack, the threat actor infects several computer systems or IoT devices with malware and converts them into a bot (zombie). The threat actor controls the infected systems remotely and sends them updated instructions to target a particular address. Figure 1 shows DDoS attack overview scenario.

Figure 1. DDoS attack overview.

Other threats: Discrepancies and contradictory controller configurations might cause network knowledge to be lost [8]. Malicious apps for controllers are simple to create.

In between controllers and the program, there really is no appropriate trust evaluation technique. DDoS really is one of the greatest serious security issues at the control and data planes [9] among some of the verified security flaws. DDoS assaults on managers are particularly harmful because of the significance of regulators in software-defined systems, and safeguarding controls from assaults is indeed a prominent priority of scientists. DDoS assaults flood the internet with traffic, using system resources and causing congestion issues. Numerous dispersed servers are being used to perform Assaults [10]. There are two phases of DDoS assaults. An intruder begins by assembling a global access infrastructure comprising millions of vulnerable machines termed as corpses, robotics, or

strike guests. The offensive server then delivers vast traffic to the target, on either attacker's order or on its own [11]. The attackers search for machines that are less vulnerable, to start the process. A DDoS assault is a physical aggression network strike that depletes computer resources, causing the system to stop working. It has the potential to eliminate a participant's accessible internet services, posing a major threat to the system. Whenever intruders send fraudulent network packets over the system, ordinary flow is slowed or even halted as a result of network resource consumption. As a consequence, the connectivity and data centers get to be congested, disrupting normal services. Malicious hackers generally target SDN due to its special attributes.

A few academics previously suggested blockchain-based methods for exchanging threat data, such as fraudulent IP addresses for blocklists, recognising security breaches at the network gateway level, and enabling Software Defined Network (SDN) nodes. However, there is indeed a gap in knowledge among network security professionals who aim to minimize DDoS potential attacks and blockchain professionals who design decentralised apps but aren't necessarily information security expertise. According to our previous art analysis, no considerable effort has been done to investigate blockchain's part in reducing DDoS attacks. As a result, we suggest this blockchain technology could be used to combat distributed denial of service assaults. Furthermore, blockchain-based technologies are classified according to their DDoS protection installation position on the network. Finally, the following are the main contribution of this manuscript:

- To suggest an Ouroboros Sharding technique for boosting the blockchain's scalability. Shard numbers and sizes are determined based on application demands, allowing users to assign resources dynamically.
- To study the effectiveness, designers collaborate with just an IT order to implement an assessment in a genuine blockchain-based SDN environment. The outcomes of the experiments show how ShChain 3D-ResNet may fare better over comparable methodologies of categorization.
- The 3D-ResNet model enables the Efficient Attention Modules (EAM) such as EAM-S where S indicates spatial and EAM-T module where T indicates temporal is included. These EAMs learn attention weights for various channels in spatial aspect, as well as attention weights for multiple frames in temporal aspect whereby imposing symmetry constraints

Research is prearranged as given here: Related works are explained in Section 2. The proposed methodology is explained in Section 3 by defining the threat model. Under Section 3.2. proposed3D-ResNet model is described. Under Section 3.3, the proposed efficient attention module is described. Under Section 3.4, Sections 3.5 and 3.6 proposed group convolution layer, reinforcement layer and detecting type of DDoS attack is given. Under Section 3.7 Construction of Sharding chain structure for privacy is given. In Section 4, results and discussion are done. Finally, the research ends with Section 5 conclusion and future scope.

2. Related Works

During the last 10 years, a DDoS assault is one of the most serious security concerns to SDN networks. This has the ability to once again prohibit legitimate users as well as using system resources, but also to absolutely ruin the network. As a result, defending the SDN network against DDoS attacks is critical.

On system log data, [12] employed an unsupervised DL network to filter out average data. Each day, an attribute mark containing an abstract of each user's organization logs is formed as well as fed into a Deep Neural Network made for every user, with desired production adapted with the next day's feature vector for all of these networks. In [13], it has created a 1D CNN end-to-end encrypted traffic classifier. In [14], an unsupervised stack auto encoder with two hidden layers was used to train a feature extractor, and then fed into a k-means clustering technique adapted with dual centroids. [15] Used a variation autoencoder, a form of autoencoder, with other ML approaches to achieve

intrusion detection. [16] used an LSTM with one-hot encode characteristics to create a botnet detector. [17] Achieved a classification accuracy of 99.1%. They also developed an instant classifier in the direction of construe folder access patterns to classify zero-day assaults appropriately. [18] Used API calls to create an autoencoder-based classifier and test multiple categorization layers. RNNs were integrated with MLP as well as logistic regression for classification in [19], which established a technique for detecting malware that employs RNNs mixed with MLP and LR. To forecast the next API call, RNN is skilled in unsupervised comportment. [20] utilized an embedding layer to analyse raw opcode data before feeding it into a CNN with 2 convolution layers, 1 max-pooling layer, a fully connected layer, then a classification layer. In [21] is an example. DistBlockNet allows all authorities inside a distributed blockchain network to also be associated, but every local area network viewpoint includes OrchApp, Manager, and Shelter components. The primary objective of the Shelter and OrchApp modules, in specific, would be to handle attacks on various levels. A traffic control analyst and network transfer elements make up Shelters. In [22] analyses a network made up of blockchain nodes, managers, and relays that must route every data towards the protected base stations. Digitally linked, sites to somehow be evaluated, & fraudulent networks are the three kinds of blockchain network. The genuine networks are those that are recognized to be permitted to join the blockchain, those unknown nodes are indeed the next kind, as well as the networks that do not have permission to link the blockchain are indeed the third category. Researchers had used the phrase 'shielded' to represent endpoints that ChainGuard is aware of. Ref. [23] proposes a deep convolutional neural network (CNN) composite structure towards DDoS detection in Distributed systems. Ref. [24] Proposes DeepGuard, an effective outlier detection approach that uses a perfectly alright road traffic surveillance approach to enhance the detectability of assaults in Software environments. Within [25], a method for detecting and mitigating Distributed Denial of Service (DDoS) as well as Portscan assaults in SDN systems can be seen (LSTM-FUZZY). The 3 components of the LSTM-FUZZY technology introduced in this study are characterisation, outlier detection, and abatement. In [26], the Recurrent Neural Network (RNN) predictor paradigm was presented as a way to properly identify and respond to assaults which are of symmetric groups such as Distributed Denial of Service (DDoS), which can cause service outages in SDN. Within [27], the Generative Adversarial Network (GAN) architecture is used to identify Assaults, and antagonistic training is used to ensure that the system fewer vulnerable to malicious examples. The suggested system comprises well-defined components that allow traffic flows surveillance via IP load flow, allowing the abnormality warning system to respond in a close moment. A detailed comparative study has been enlisted in Table 1.

Table 1. Comparative study of existing methods.

Author Year	Proposed Method	Merits	Demerits
Tuor et al., (2019) [12]	unsupervised DL network	It can detect attacks in early stages	Overload the controller
Kobojek et al., (2020) [13]	1D CNN	Low false positive and false negative	Only handle on type of DDoS attack
Maimó et al., (2018) [14]	unsupervised stack auto encoder	Effectively protect the controller from attack	Delay when handling enormous number of incoming packets
Abdulhammed et al., (2018) [15]	autoencoder	Able to distinguish attack traffic from legitimate traffic	Only handles low traffic rate and threshold is fixed
Diro et al., (2018) [16]	LSTM	Low false positive	Less accuracy
He et al., (2017) [17]	CNN	Traces attack score reduces workload on controller in early stage	Works after controller received traffic flow which leads to flooding of packets

Table 1. *Cont.*

Author Year	Proposed Method	Merits	Demerits
Kobojek et al., (2016) [18]	autoencoder-based classifier	Reduce overload between controller and switches	Unable to separate the legitimate user
Lotfollahi et al., (2017) [20]	RNNs mixed with MLP	Able to distinguish the attack from flash crowd	Overload the controller
Cox et al., (2015) [19]	RNN	Reduces controller overhead	Delay in DDoS attack detection
Sharma et al., (2017) [21]	DistBlockNet	Quick reaction in detecting DDoS attack	Unable to deal with complex traffic flow in the switch
Steichen et al., (2017) [22]	ChainGuard	Reduces resource consumption	Requires high computing resources and processing power of SDN controller
Haide et al., (2020) [23]	convolutional neural network (CNN)	High detection ratio on DDoS attack	Less accuracy
Phan et al., (2020) [24]	DeepGuard	Prompt, versatile, and accurate detection of DDoS attack	High resource consumption on controller
Novaes et al., (2020) [25]	LSTM-FUZZY	Limits false positive and false negative rate	Does not care about the temporal characteristic in order to accelerate detection process
Polat et al., (2022) [26]	Recurrent Neural Network (RNN)	Reduces congestion of incoming packets at controller	Need time to detect the attack
Novaes et al., (2021) [27]	Generative Adversarial Network (GAN)	Increased detection accuracy	Difficult to detect unknown or new types of DDoS attack

Even so, the review does not wrap conventional systems that use neural network objective and accurate on ideal process position to prevent and identify DDoS attacks in the virtual environment, as well as a detailed account of DDoScyberattacksBlockchain systems to safeguard decentralised connections [28]. Our goal with this project is to examine the link between network security researchers and the blockchain development sector, allowing the researchers to have this article as a point of comparison while they pursue their study into leveraging blockchain solutions in SDN.

3. System Model

As DDoS attacks become more common and their methods become more varied, the requirement for designed for efficient to effectively deflect the onslaught grows. It is important to remember, however, that only customer-ASes collaboration is an extra way to establish defense mechanisms. The structure shown in Figure 2 is made up of three parts:

- Customers can use decentralized applications to submit white or blacklisted IPs to the Sharding ring ouroborusblockchain.
- ASes: can broadcast white or blacklisted IPs, get lists including the published Proxy servers, and use DDoS countermeasures.
- Blockchain/Smart Contract: the public Sharding ring ouroborusblockchain is powered by Structural rigidity smart contracts, including the logic for reporting IPs to the blockchain.

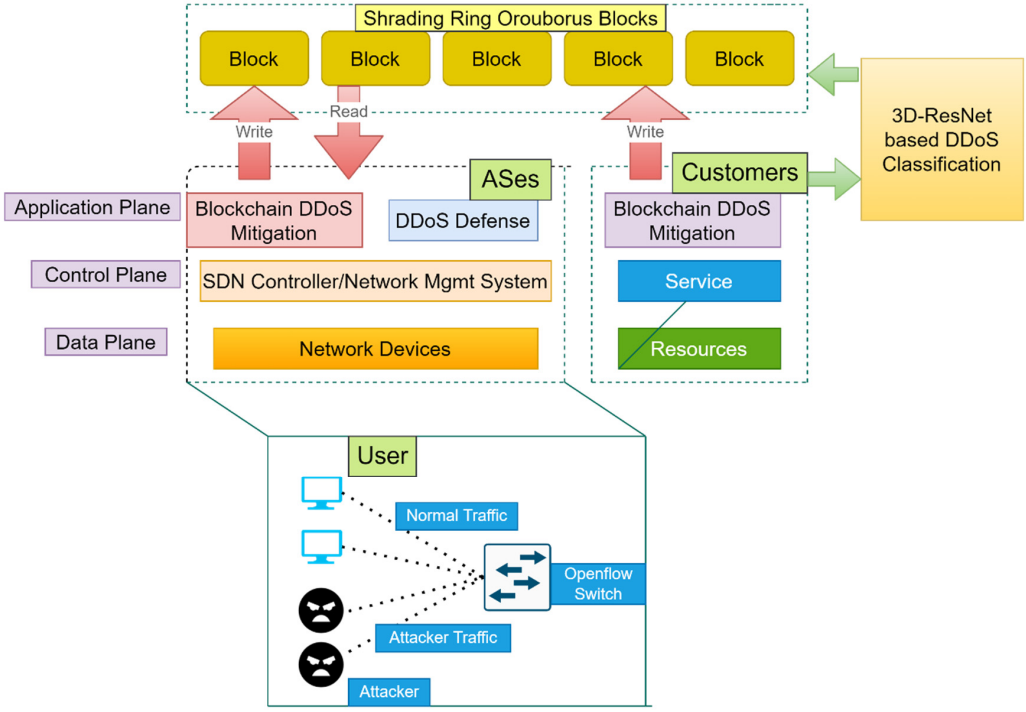


Figure 2. SDN architecture and occurrence of DDoS attack.

3.1. Threat Model:DDoS Attack

Explore a scenario in which N users, each with their data x_i and a one-hot vector of labels Yi, collectively train a neural network (NN) model. A querier—which might be one of the N parties or an external entity—questions the model at the end of training phase as well as obtains prediction results data y_q based on its evaluation data x_q. The parties involved in the training process want their local data privacy, intermediate model updates, and the final model to be preserved. The querier collects prediction results from the trained model while maintaining the confidentiality of its evaluation data. Assume that the parties are interconnected and structured in a tree-network structure for communication efficiency [28]. However, because our system is entirely dispersed and makes no assumptions about hierarchy, it is agnostic to network architecture, allowing us to explore a fully connected network. Explore a passive-adversary model with up to $N-1$ SDN users colluding to extract information about the other parties' inputs through membership inference.

The attacker's goal would be to cause genuine users' transmissions to be dropped or delayed, causing the channel's efficiency to degrade therefore, eventually, the consumers' enjoyment to be ruined. A "table-miss" occurrence is normally raised whenever a new flow of packets hits a switch. As a result, a Protocol In notification is sent to the controller, seeking action. The switch performs the determined operation to all following packets in the same flow without even any controller interaction. While waiting for the controller to respond, the switch buffers the incoming packets in its buffer memory. In the event that the buffer fills up, successive packets are missed due to a lack of buffer capacity [29].

3.2. 3D ResNet Architecture for DDoS Attack Classification

The attack using single-pixel and pattern backdoors is implemented by visualising the convolutional filters in the first layer of the 3D-ResNet, as shown in Figure 3. The 3D-ResNet appears to have learned convolutional filters for detecting backdoors. The presence of dedicated backdoor filters shows that backdoors are sparsely written in the 3D-deeper ResNet's layers. An input, convolution layer 1, 2, embedded layer, pooling layer 1, 2, and full connection layer 5 make up this system. Returns a high-dimensional package vector to the reinforcement section after receiving a preprocessed and feature extracted file. The REIN layer 6, REIN layer 7, complete connection layer 8, and OUTPUT layer make up

the reinforcing portion. It can take a set of HD package vectors as well as turn them into a vector that indicates the likelihood that session belongs to every class.

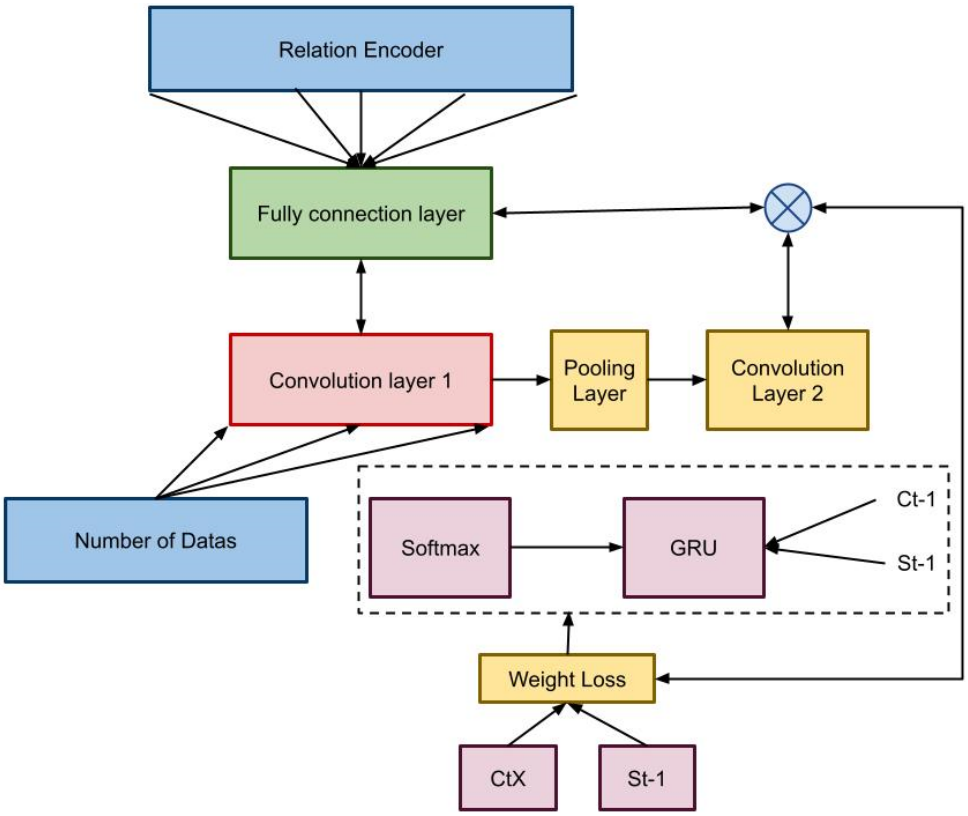


Figure 3. Architecture of 3D-ResNet for classifying.

The Softmax layer outputs classification's final result based on the probability vector. The initial step in REIN layer is to figure out what data from the cell state model will discard. Forgetting gate is used to make this decision. Gate reads $h(t-1)$ and $x(t)$ and gives every number in $C(t-1)$ cell state a value between 0 and 1; 1 signifies "fully retained," while 0 means "totally discarded." W and b stand for weight and bias in NN.

$$f(t) = \mu[Wf.[h(t-1), x(t)] + b(f)] \tag{1}$$

Next step is to figure out how much additional data to include in the cell state. A sigmoid layer is used first to determine which data needs to be updated. As an alternative to updating, a layer generates a vector. The two portions are then concatenated to update the cell's status as needed.

$$I(t) = \mu[Wi.[h(t-1), x(t)] + b(i)] \tag{2}$$

$$C(t) = tan[h(Wc.[h(t-1), x(t)\,] + b(c)] \tag{3}$$

$$C(t) = f(t) \times C(t-1) + i(t) \times C(t) \tag{4}$$

Output gate finds the cell's output. A sigmoid layer decides which sections of the cell are in which condition first. The cell state is then multiplied by the output of the sigmoid gate after going through a function to get a value between -1 and 1. As a result, the output is evaluated as follows:

$$O(t) = \mu[W(o).[h(t-1), x(t)] + b(o)] \tag{5}$$

$$H(t) = O(t) \times tan\, h\, (Ct) \tag{6}$$

The first few data stream packets can establish connections; such packets can be found in both standard and attack data streams. Long payloads, as well as attack data, may be included in the subsequent few frames. Introduced class weights to solve the classification issue: A more considerable class weight indicates that the class is more important. More examples are classified into high-weight groups compared to the case when weight is not taken into account. Equation (7), where Wi represents class weight of class I and ni represent volume of traffic in class I is used to computing the class weight.

$$Wi = \frac{\sum_{i=0}^{k-1} ni}{ni} \tag{7}$$

When the model is trained, the weighted loss function directs the model's attention to samples from underrepresented groups. In this model, k is number of categories, y is label, and p is neural network's output, which is the probability that the methods predict that type is I and is calculated via Softmax.

3.3. Proposed Efficient Attention Module (EAM)

It is made up of two modules: an EAM-S module that studies attention weights for various channels in spatial dimension as well as an EAM-T module that learns attention weights for various frames in temporal dimension. The 4D cost volume $V \in R^{H \times W \times T \times C}$ is sequentially passed into these two modules. Processing flow of EAM is given as $V' = M_c(V) \times V, V' = T_{trans}(M_c(V)$ and $V''' = M_t(V_1) \times V', V'' = T_{trans}(t(V') \times V')$ and $V''' = V + V''$. The function of training ResNet vector is applied to the accurate unit's v as shown below.

$$F(v,h) = -\sum_{p=1}^{p}\sum_{q=1}^{q} w(pq)v(p)h(q) - \sum_{p=1}^{q} \propto (p)v(p) - \sum_{q=1}^{q} \beta(q)h(q) \tag{8}$$

Maximization of Equation (1) is as follows. $w(pq)$ is programmatic interaction among visible unit $v(p)$ and hidden unit $h(q)$, α, β are bias terms, and p, q are the amount of visible and hidden units. A training vector's following log possibility is concerned with the weight of inconsistency. Hidden units from ResNet do not provide a direct response that aids in the creation of an ideal unbiased sample from $(v(p),\ h(q))$ data. The updating procedure is described as a difficult stage.

$$\mu W(pq)\ v(p)h(q)\text{data} - v(p)h(q)\text{reconst} \tag{9}$$

The Attention Module has been well-trained at this point. The diverging Attention Module is built on top of a multilayer model frame. Furthermore, a unique Attention Module that has been stacked can be defined.

3.4. Dense Convolution Process

Let us use uhv_j in the source attention layer with softmax, as well as the prediction vector $uhv_{j|\tilde{i}}$ for viewing basic features to transfer, which is obtained by multiplying uhv_j by the transformation matrix W_j.

$$uhv_{j|\tilde{i}} = uhv_i \times W \tag{10}$$

b_{ij} is set to zero and adjusted with the a_{ij} scale agreement in the "softmax routing" feature. On a scale of 1 to 10, its agreement a_{ij} is determined.

$$a_{ij} = uhv_{j|i} \tag{11}$$

$$b_{ij} = b_{ij} + a_{ij} \tag{12}$$

This layer aims to create a given and aggregated abstract vector by agreeing that each output will be utilised as an input. At the focus layer, the following procedure occurs:

$$a_{ij} = \frac{\exp(ei)}{\sum k \exp(ek)} \tag{13}$$

$$ei = (q, hv_i) \tag{14}$$

$$(q, uhv_i) = qT \times uhv_i \tag{15}$$

The q, in this case, is a trainable pattern vector. After that, calculate and provide the weighted total to the application classification overall target length as a set long attention vector.

$$o = \Sigma_i a_i uhv_i \tag{16}$$

Once the intrusions in the collected data were identified, the secure data transmission to the cloud server commenced.

3.5. Reinforcement Layer

Convolution using sparse kernels was employed in group convolution, which allowed the parameters to be reduced. For example, knowing that doing origin convolution on figure maps with form $1 \times C \times w \times h$ with k number of kernels with shape k_w, k_h, C takes N_{con}:

$$N_{con} = (w - k_w + 1) \times (h - k_h + 1) \times k_w \times k_h \times C \times k \tag{17}$$

It is also discovered that the calculation of group convolution is significantly reduced. This ReLU will be followed if g is groups and k' indicates number of kernels were employed.

$$C = g \times k' \tag{18}$$

And computation of group Recurrent N_{GC} is as follows:

$$N_{GC} = (w - k_w + 1) \times (h - k_h + 1) \times k_w \times k_h \times k' \times g \tag{19}$$

The computation of N_{con} is determined to be k times larger than that of N_{GC}. Zero fields of kernels join calculation in group convolution; therefore, the advantage of group convolution may be less clear. Because convolution only works in a local region, the output feature map has difficulty obtaining enough information to determine the link between channels. Due to the limited receptive field at the front layer of the network, this problem becomes more serious. Squeeze operation entails converting a channel's entire spatial feature into a global feature, which is accomplished using global average pooling. Squeezing calculation is shown in the equation below.

$$z_c = F_{sq}(u_c) = \frac{1}{j \times w} \sum_{i=1}^{j} \sum_{j=1}^{w} u_c(i, a) \tag{20}$$

u_c is input of feature maps, while j and w are height and breadth of feature maps. After we obtained the global feature, we would need to perform another operation called excitation to learn the relationship between channels. Excitement should fulfill two requirements: first, it is adaptable. It should also be able to learn nonlinear channel relationships. Second, the learning relationship isn't mutually exclusive because various channel features can be used instead of one hot form. There is an equation based on the criteria.

$$S = F_{ex}(z, W)\sigma(g(z, W)) = \sigma(W_2 \partial(W_1 z)) \tag{21}$$

where ∂ is the ReLU function, $W_1 \in RC2/r$ and $W_2 \in RC2/r$ are trainable parameters and z is input feature maps. A bottleneck structure with two recurrent layers minimises the

model's complexity and increases its generalisation capabilities. To lower the dimension, the first convoluted recurrent layer is employed. The final FC layer returns the size to its original state.

3.6. Detection of DDoS Attack Type

Teardrop attack: this attack requires the "Teardrop.c" software to send incorrect redundant contents of IP chunks in TCP packet headers. As a consequence, even during re-assembly procedure, the victim's virtual server will collapse. Algorithm 1 explains type of DDoS attack detection process.

Algorithm 1: DDoS Attack type Detection

Input: Dataset (D), Attack types (A), weight matrix W,
hidden layer element bias B, and visible layer element bias A
Output: Categorized DoS attack
for all k < D
do
initialize input-hidden weight matrix W
Randomly initialize hidden biases B
For k = 1 to N do
for D = 1 to l do
Update {Wi, Bi, Ni}
End for
End for
Weight of class i
$wi = \frac{\sum_{i=0}^{k-1} ni}{ni}$
Update wi
if wi = Bi then, DoS attack detected and categorize it
if wi ≠ Bi then, no occurrence of attack

Buffer overflow attack: the attacker creates the vulnerable target malware in order to take advantage of a buffering overflowing flaw.

The "Land.c" programme is used to transmit forged TCP SYN packets with the victim's IP address in the input and output fields.

Attacker sends an IP packet that exceeds the IP protocol's size limit, resulting in a ping of death.

Smurf attack: an inside cloud-based DoS attack might be carried out by leveraging a compromised cloud system device as an intermediary to send echo requests to local broadcast IPs.

3.7. Construction of Sharding Chain Structure for Privacy

The ShChain architecture, which blends 3D-ResNet block diagram with some elements of blockchain method has been proposed as shown in Figure 4. Each block contains the following data, which aids in block authentication against tampering.

Table 2 indicates the block number for possible identification. In the current investigation, it has no bearing. AShChain block is depicted in Figure 3. It contains the hashes of current as well as previous blocks, the current layer's public and private keys, public keys of layers appearing immediately before and after current layer and next layer's AES key as well as model parameters. Blocks in metaring, like blocks in a blockchain, have a shared common ledger that records model's state. Previous block's hash, as well as current and next layer's parameters, determine the hash associated with a block. The hash of a block j in the 3D-ResNet architecture that corresponds to layer I is:

$$Hash_j = \Omega\left(Hash_{j-1}, params_i, params_{i+1}\right) \tag{22}$$

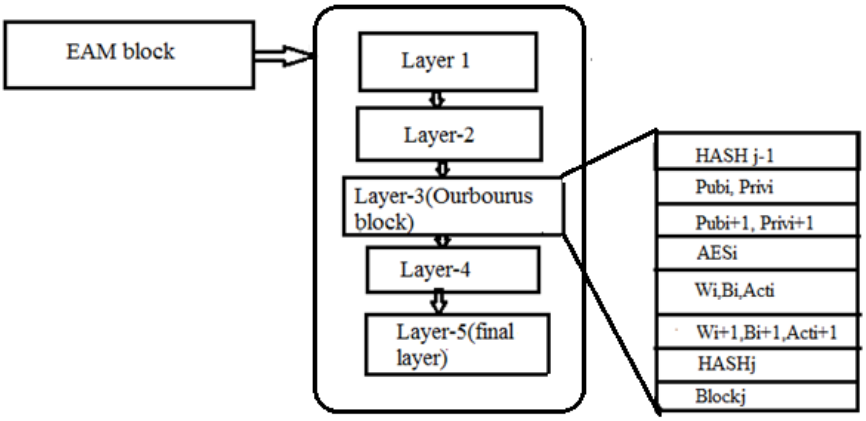


Figure 4. Block diagram of ShChain [30].

Any acceptable hash function Ω such as SHA256 can be used here. The ouroboros block stores hashes of all the blocks after the model is successfully established in the metaring framework. This information is later utilised to track compromised blocks in a tampering attack. One thing to keep in mind is that the block's in-ring are not in any particular order. They're organised in random order. Even blocks themselves are unaware of layer's sequential index, which they serve. Ouroboros block is beginning as well as finish of any 3D-ResNet query. It is ring's only known block.

$$Authenticity \leftarrow origHASHprev = HASHprev \tag{23}$$

The Ouroboros block has two modes of operation:

- Query Mode (*authenticity* = TRUE): This is standard mode, which accepts input as well as returns output.
- Tracking Mode (*authenticity* = FALSE): When the block parameters of any block in network change, this mode is triggered. Any change in a parameter is propagated to hash of previous block. Block suspends general query business in this mode as well as attempts to identify compromised blocks.

Hash of ouroboros block is a function of only layer parameters: $Hash_{\text{ouroboros}} = \Omega(params_{\text{ouroboros},}params_{1,})$. Because the ouroboros block has outstanding obligations, it should be filled by someone you can trust, such as the model's owner. Every Metaverse user creates their unique ID based on the logic and distance between nodes.

$$d_i(H_i, H_j) = \frac{\sum_{p,q\in\{Hi\cup H_j - H_i\cap H_j\}}\left(x_{pq}^{H_i} + x_{pq}^{H_j}\right)}{\sum_{pq\in H_i\cup H_j}\left(x_{pq}^{H_i} + x_{pq}^{H_j}\right)} \tag{24}$$

Equation (25) shows a randomised technique for two Metaverse user nodes to protect meta world privacy in a decentralised manner. A(R) ∈ S attains data privacy.

$$\text{Hr}[\mathcal{A}(R) \in S] \leq \exp(\epsilon) \cdot \text{Hr}\left[\mathcal{A}\left(R'\right) \in O\right] \tag{25}$$

$$\vec{m_i} = m_i + \text{Laplace}(s/\epsilon) \tag{26}$$

where s denotes the degree of sensitivity, and

$$s = \max_{H,H'} \|f(H) - f\left(H'\right)\|_1 \tag{27}$$

To protect privacy of Metaverse data, every information is encrypted as well as signed with public and private keys $(\text{PK}_\text{i}, SK_i)$. Then, all transactions are calculated using

MAE(M), and H_j is broadcast. The record is stored on distributed ledger if all transactions are approved. The formula for MAE is as follows:

$$\begin{aligned} MAE(m_i) &= \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)| \\ MAE(H_j) &= \gamma \cdot MAE(m_j) + \frac{1}{n}\sum MAE(m_i) \end{aligned} \tag{28}$$

To protect data, current methods use encryption. However, because of specific security attacks, data providers are in danger of sharing personal data. A straightforward approach is to send data to requester with valid details while protecting privacy of the data owners. Instead of giving raw data, data providers send the requester learned models.

Table 2. Notations and expansions.

Notations	Expansion
$Hash_j$	Hash of *j*th block
Pub_i	public key of *i*th block
Priv_i	private key of *i*th block
AES*i*	AES key of *i*th block
w_i	weigh
b_i	bias
act_i	activation function

4. Result and Discussion

Experimentations were carried out on two publicly available datasets to validate the effectiveness of the proposed scheme such as:

Dataset-1: LID-DS is a HIDS-based IDS Dataset proposed by Martin et al. from Leipzig University [29]. This dataset comprised current attack scenarios and was created using Ubuntu Linux 18.04. The collection contains traces of both regular and assault activity for each design. It is the most recent IDS dataset and includes attack scenarios from the last few months.

Dataset-2: Gideon et al. [30] suggested and announced a public release of the ADFA22 IDS dataset, which is distinct from the current legacy dataset. It can be used to put HIDS in place. Only system call numbers are sequenced in the trace file.

4.1. Experimental Setup

The structure of our private blockchain network is linear. In this case, the laptop serves as a mining node with sufficient processing capacity to mine continually. The RPI 3b+, on the other hand, is a data node that focuses on data sharing. Ping commands can be used to assess RTT values between two devices 100 times because blockchain sends messages across a network connection. The average RTT is 5.047 milliseconds, implying that a transaction takes 4 milliseconds to process and 5 milliseconds to send to next node.

4.2. Comparative Analysis

Experimental result was carried out using the parameters encryption time, decryption time, accuracy, precision, recall, F1-score. These parameters were compared with existing methods such as Convolution Neural Network (CNN), DeepGuard, LSTM-Fuzzy, RNN, Generative Adversarial Network (GAN) with proposed ShChain_3D-ResNet.

Precision: Commonly used in two-class problems which measure a particular class. It is a negative class propagation which is predicted negatively and is called a true negative rate. The formula is:

$$precision = \frac{TN}{TN + FP} \tag{29}$$

Recall: Classification model ability measures certain selective class instances from the dataset. It is a positive propagation and the prediction is a positive one. The formula used is:

$$recall = \frac{TP}{TP + FN} \tag{30}$$

F1-score is utilized to determine the prediction performance. It is the weighted average of precision and recall. The value of 1 determines the best while 0 the worst. F1-score does not consider TNs and is calculated as:

$$f1 - Score = \frac{2 \times P \times R}{P + R} \tag{31}$$

The encryption time is indeed the time required for a data encryption to generate a cypher text from plaintext. The efficiency of data encryption is calculated using encryption time. It denotes the encryption speed. Comparison between existing and proposed for dataset-1 and dataset-2 have been listed in Tables 3 and 4 respectively. Results obtained from 3D-ResNet runtime analysis is shown in Table 5. Table 6 shows outcome analysis on ouroboros Sharding blockchain model. The mean and standard deviation of Shchain_3D-ResNet obtained from experiments are listed Table 7.

Decoding is the process of returning user information to its original form once it has been deemed inaccessible via cryptography.

Table 3. Comparison between existing and proposed for dataset-1.

Methods	Accuracy	Precision	Recall	F1-Score	Encryption Time (ms)	Decryption Time (ms)
CNN	67.4	73.5	43.5	48.3	56.3	82.7
DeepGuard	58.2	77.5	56.4	35.2	77.8	85.1
LSTM-Fuzzy	83.4	63.2	71.3	46.2	73.2	89
RNN	76.2	58.3	65.2	78.1	56.2	73.5
GAN	84.1	82.4	69	46.2	77.2	77.1
ShChain_3D-ResNet	95.6	97.3	95.2	94.4	32.5	35.2

Table 4. Comparison between existing and proposed for dataset-2.

Methods	Accuracy	Precision	Recall	F1-Score	Encryption Time (ms)	Decryption Time (ms)
CNN	67.2	57.4	74.3	85.3	45.6	51.3
DeepGuard	86.3	48.2	75.4	81.4	43.6	55.2
LSTM-Fuzzy	77.4	55.6	79.5	79.5	49.6	40
RNN	74.3	71.4	73.1	74.6	46.3	42.3
GAN	81.3	78.3	75.3	83.4	47.5	44.9
ShChain_3D-ResNet	97.3	95.3	96.1	98.2	32.1	36.2

Table 5. Runtime of 3D-ResNet.

Number of Nodes	3D-ResNet Runtime
100	6.5
200	7.2
300	8.2
400	8.1
500	7.4

Table 6. Analysis on ouroboros Sharding blockchain.

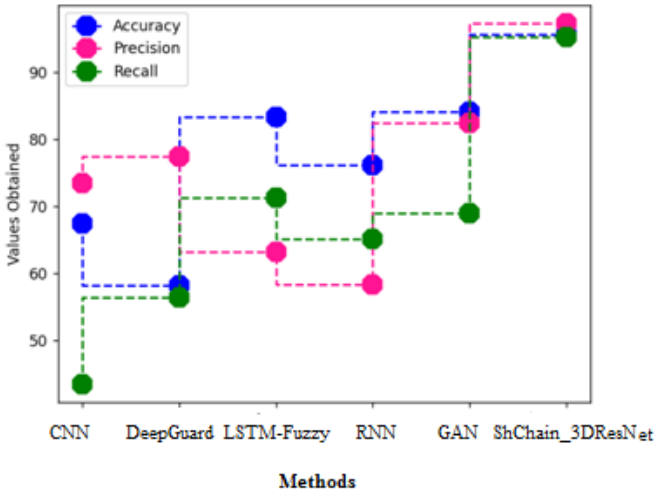
Number of Blocks	Learning Iteration	Learning Time (s)	Learning Error	Mining Iteration	Mining Time	Mining Error
1	245.2	134.5	34%	34.5	121.31	32%
2	192.1	120.5	32%	32.1	132.2	30%
3	170.3	121.4	31%	38.3	131.4	31%
4	165.3	131.5	30%	36.2	146.2	32%

Table 7. Mean and standard deviation of Shchain_3D-ResNet.

Number of Neurons	Mining Iteration	Mining Time	Mean ± Standard Deviation	Mining Threshold
60-4-60	34.5	121.31	34.3 ± 32.4	145
124-4-60	32.1	132.2	34.2 ± 35.6	150
188-4-60	38.3	131.4	32.5 ± 36.7	165
252-4-60	36.2	146.2	32.5 ± 35.7	170

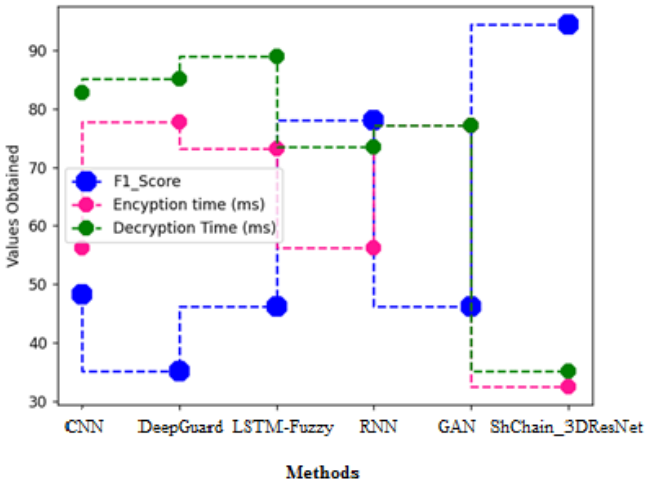
5. Discussion

Figures 5 and 6 illustrate the comparison of encryption time, decryption time, accuracy, precision, recall, F1-score between existing Convolution Neural Network (CNN), DeepGuard, LSTM-Fuzzy, RNN, Generative Adversarial Network (GAN) and proposed ShChain_3D-ResNet where the x-axis shows various methods and y axis indicates parameters in percentage and ms. When analyzing, the ShChain _3D-ResNet achieves 95.6% accuracy, 97.3% precision, 95.2% recall, 94.4% F1-score, and 32.5 ms of encryption time 35.2 ms of decryption time for dataset-1. Furtherly, ShChain_3D-ResNet achieves 97.3% accuracy, 95.3% precision, 96.1% recall, 98.2% F1-score, 32.1 ms of encryption time, and 36.2 ms of decryption time for dataset-2. Figure 7 illustrates the comparison of Runtime of 3D-ResNet, where the x-axis indicates number of nodes and y axis indicates time in ms. It is notable that for 100 nodes, it achieves 6.5 ms, 200 number of nodes it achieves 7.2 ms, 300 number of nodes 8.2 ms, 400 number of nodes it is 8.1 and 500 number of nodes 7.4 ms of run time is achieved. Figures 8 and 9 show the Analysis of ouroboros Sharding blockchain by fixing the number of blocks as 4, which has the better iteration rate of 245.2 for block-1 with 134.5 s. The error is considerably less in block number 3, attaining 31%. For the mining process in the blockchain network, the iteration is best for block number 3 by achieving 38.3 with time 131.4 s. The error for the mining process is less for block number 2 by achieving 30%. From Table 7 it is analyzed that ShChain_3D-ResNet achieves approximately 34.2 ± 35.6 of mean and standard deviation.



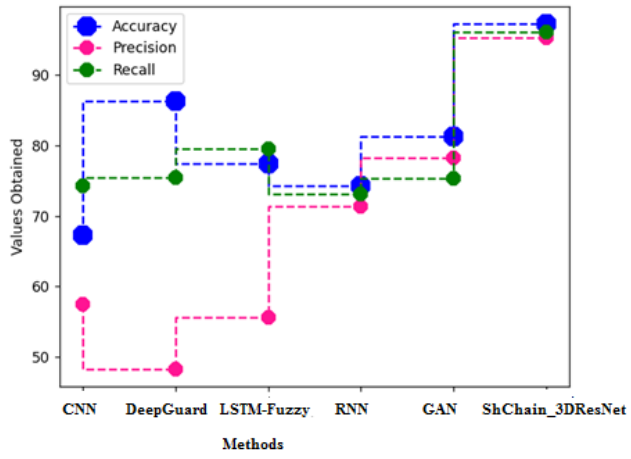
(**a**)

Figure 5. *Cont.*

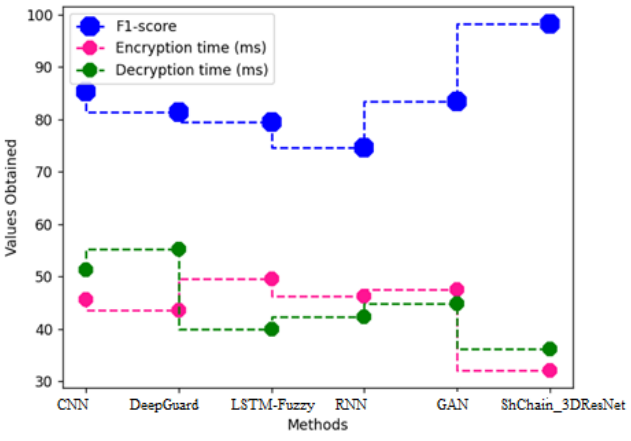


(**b**)

Figure 5. Analysis of existing and proposed method for dataset-1. (**a**); Accuracy, Precision and Recall comparison; (**b**) F1-score, encryption and decryption time analysis.



(**a**)



(**b**)

Figure 6. Analysis of existing and proposed method for dataset-2. (**a**) Performance analysis of NN models; (**b**) Encryption and decryption time analysis.

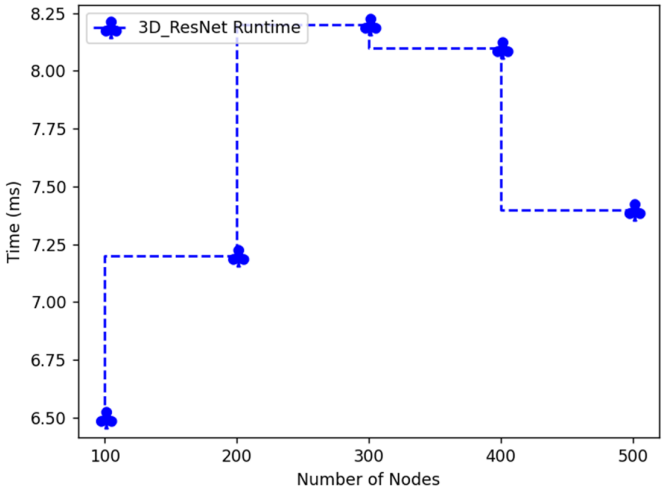


Figure 7. Comparison of runtime.

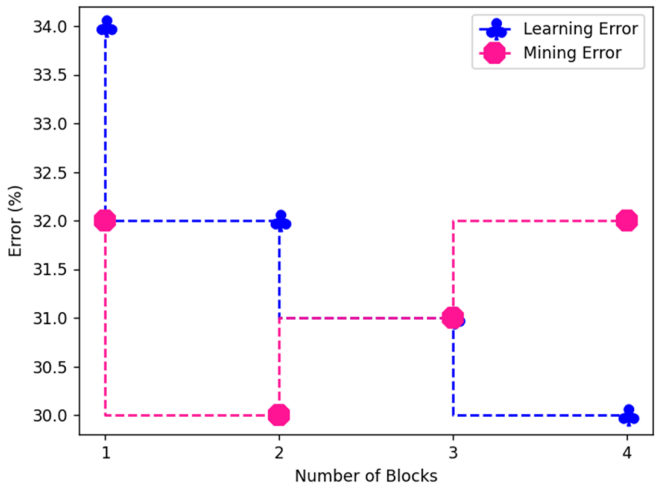


Figure 8. Analysis of errors.

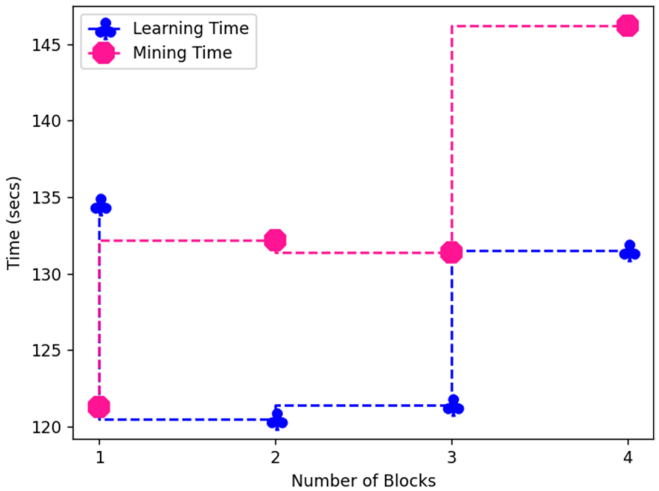


Figure 9. Analysis of time.

6. Conclusions

In this research, ShChain 3D-ResNet, a revolutionary blockchain-based architecture for effectively managing SDN applications, was presented. ShChain makes use of the benefits of blockchain as well as Sharding technology to enable smart as well as trustworthy interactions between customers and ASes. Furthermore, we suggested ouroboros, a new Sharding strategy that can increase ShChain performance while simultaneously utilising Multi User (MU) resources for SDN. Finally, we conducted experiments to calculate the significance of our proposed method by comparing with state-of-the-art techniques and found that the proposed ShChain_3D-ResNet achieved a 95.6% accuracy, 97.3% precision, 95.2% recall, 94.4% F1-score, 32.5 ms encryption time and a 35.2 ms decryption time for dataset-1. Furthermore, it achieved a 97.3% accuracy, 95.3% precision, 96.1% recall, 98.2% F1-score, 32.1 ms encryption time, and 36.2 ms decryption time for dataset-2. For future research, game method is extended to a multi-leader-multi-follower ouroboros Sharding method.

Author Contributions: Conceptualization, E.F. and P.M.K.; methodology, E.F.; software, E.F.; validation, P.M.K.; formal analysis, E.F.; resources, P.M.K.; writing—original draft preparation, E.F.; writing—review and editing, E.F.; visualization, E.F.; supervision, P.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Al-Adaileh, M.A.; Anbar, M.; Chong, Y.W.; Al-Ani, A. Proposed statistical-based approach for detecting distribute denial of service against the controller of software defined network (SADDCS). In *MATEC Web of Conferences*; EDP Sciences: Les Ulis, France, 2018; Volume 218, p. 02012.
2. Sanjeetha, R.; Srivastava, S.; Pokharna, R.; Shafiq, S.; Kanavalli, A. Mitigation of DDoS attack instigated by compromised switches on SDN controller by analyzing the flow rule request traffic. *Int. J. Eng. Technol.* **2018**, *7*, 46–49.
3. Mladenov, B.; Iliev, G. Searching for optimal software defined network controller against dDoS attacks. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 20–22 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–4.
4. Fenil, E.; Kumar, P.M. Towards a secure software defined network with adaptive mitigation of dDoS attacks by machine learning approaches. In Proceedings of the 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 28–29 January 2022; pp. 1–13.
5. Zhao, Y.; Iannone, L.; Riguidel, M. On the performance of SDN controllers: A reality check. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 79–85.
6. Nanda, S.; Zafari, F.; DeCusatis, C.; Wedaa, E.; Yang, B. Predicting network attack patterns in SDN using machine learning approach. In Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 7–10 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 167–172.
7. Fenil, E.; Kumar, M.P. Survey on DDoSdefense mechanisms. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5114. [CrossRef]
8. Gkountis, C.; Taha, M.; Lloret, J.; Kambourakis, G. Lightweight algorithm for protecting SDN controller against DDoS attacks. In Proceedings of the 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC), Valencia, Spain, 25–27 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
9. Dong, P.; Du, X.; Zhang, H.; Xu, T. A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
10. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
11. Bouras, C.; Kollia, A.; Papazois, A. SDN & NFV in 5G: Advancements and challenges. In Proceedings of the 2017 20th Conference on innovations in clouds, internet and networks (ICIN), Paris, France, 7–9 March 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 107–111.
12. Tuor, A.; Kaplan, S.; Hutchinson, B.; Nichols, N.; Robinson, S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *arXiv* **2019**, arXiv:1710.00811.
13. Kobojek, P.; Saeed, K. Application of recurrent neural networks for user verification based on keystroke dynamics. *J. Telecommun. Inf. Technol.* **2020**, *3*, 80–90.

14. Maimó, L.F.; Gómez, A.L.P.; Clemente, F.J.G.; Pérez, M.G. A self-adaptive deep learning-based system for anomaly detection in 5G networks. *IEEE Access* **2018**, *6*, 7700–7712. [CrossRef]
15. Abdulhammed, R.; Faezipour, M.; Abuzneid, A.; AbuMallouh, A. Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE Sens. Lett.* **2018**, *3*, 1–4. [CrossRef]
16. Diro, A.A.; Chilamkurti, N. Leveraging LSTM Networks for Attack Detection in Fog-to-Things Communications. *IEEE Commun. Mag.* **2018**, *56*, 124–130. [CrossRef]
17. He, Y.; Mendis, G.J.; Wei, J. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Trans. Smart Grid* **2017**, *8*, 2505–2516. [CrossRef]
18. Rybnik, M.; Panasiuk, P.; Saeed, K.; Rogowski, M. Advances in the keystroke dynamics: The practical impact of database quality. In *Computer Information Systems and Industrial Management*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7564, pp. 203–214.
19. Cox, J.A.; James, C.D.; Aimone, J.B. A signal processing approach for cyber data classification with deep neural networks. *Procedia Comput. Sci.* **2015**, *61*, 349–354. [CrossRef]
20. Lotfollahi, M.; Shirali, R.; Siavoshani, M.J.; Saberian, M. Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *arXiv* **2017**, arXiv:1709.02656. [CrossRef]
21. Sharma, P.K.; Singh, S.; Jeong, Y.-S.; Park, J.H. DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks. *IEEE Commun. Mag.* **2017**, *55*, 78–85. [CrossRef]
22. Steichen, M.; Hommes, S.; State, R. ChainGuard—A firewall for blockchain applications using SDN with OpenFlow. In Proceedings of the International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm), Chicago, IL, USA; 2017; pp. 1–8.
23. Haider, S.; Akhunzada, A.; Mustafa, I.; Patel, T.B.; Fernandez, A.; Choo, K.K.R.; Iqbal, J. A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *IEEE Access* **2020**, *8*, 53972–53983. [CrossRef]
24. Phan, T.V.; Nguyen, T.G.; Dao, N.N.; Huong, T.T.; Thanh, N.H.; Bauschert, T. DeepGuard: Efficient anomaly detection in SDN with fine-grained traffic flow monitoring. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1349–1362. [CrossRef]
25. Novaes, M.P.; Carvalho, L.F.; Lloret, J.; Proenca, M.L. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access* **2020**, *8*, 83765–83781. [CrossRef]
26. Polat, H.; Türkoğlu, M.; Polat, O.; Şengür, A. A novel approach for accurate detection of the DDoS attacks in SDN-based SCADA systems based on deep recurrent neural networks. *Expert Syst. Appl.* **2022**, *197*, 116748. [CrossRef]
27. Novaes, M.P.; Carvalho, L.F.; Lloret, J.; Proença, M.L., Jr. Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments. *Future Gener. Comput. Syst.* **2021**, *125*, 156–167. [CrossRef]
28. Leipzig Intrusion Detection Dataset. Available online: https://www.exploids.de/lid-ds/ (accessed on 18 April 2020).
29. WEKA Tool. Available online: https://www.cs.waikato.ac.nz/ml/weka/ (accessed on 1 February 2020).
30. Nguyen, C.T.; Hoang, D.T.; Nguyen, D.N.; Dutkiewicz, E. MetaChain: A Novel Blockchain-based Framework for Metaverse Applications. *arXiv* **2021**, arXiv:2201.00759.