sustainability

Article

Applying Transfer Learning Approaches for Intrusion Detection in Software-Defined Networking

Hsiu-Min Chuang [1,*] and Li-Jyun Ye [2]

[1] Department of Information and Computer Engineering, Chung Yuan Christian University, Taoyuan City 320, Taiwan
[2] Department of Computer Science and Information Engineering, Chung Cheng Institute of Technology, National Defense University, Taoyuan City 335, Taiwan; tomatoccit@gmail.com
* Correspondence: showmin@cycu.edu.tw

Abstract: In traditional network management, the configuration of routing policies and associated settings on individual routers and switches was performed manually, incurring a considerable cost. By centralizing network management, software-defined networking (SDN) technology has reduced hardware construction costs and increased flexibility. However, this centralized architecture renders information security vulnerable to network attacks, making intrusion detection in the SDN environment crucial. Machine-learning approaches have been widely used for intrusion detection recently. However, critical issues such as unknown attacks, insufficient data, and class imbalance may significantly affect the performance of typical machine learning. We addressed these problems and proposed a transfer-learning method based on the SDN environment. The following experimental results showed that our method outperforms typical machine learning methods. (1) our model achieved a F1-score of 0.71 for anomaly detection for unknown attacks; (2) for small samples, our model achieved a F1-score of 0.98 for anomaly detection and a F1-score of 0.51 for attack types identification; (3) for class imbalance, our model achieved an F1-score of 1.00 for anomaly detection and 0.91 for attack type identification. In addition, our model required 15,230 seconds (4 h 13 m 50 s) for training, ranking second among the six models when considering both performance and efficiency. In future studies, we plan to combine sampling techniques with few-shot learning to improve the performance of minority classes in class imbalance scenarios.

Keywords: transfer learning; meta-learning; intrusion detection; software-defined networking

check for updates

Citation: Chuang, H.-M.; Ye, L.-J. Applying Transfer Learning Approaches for Intrusion Detection in Software-Defined Networking. *Sustainability* **2023**, *15*, 9395. https://doi.org/10.3390/su15129395

Academic Editor: Zubair Baig

Received: 16 April 2023
Revised: 5 June 2023
Accepted: 7 June 2023
Published: 11 June 2023

1. Introduction

The rapid development of semiconductors and network-side information technology has paved the way for the new generation of network technology, which is moving toward software-defined networking (SDN) and network function virtualization (NFV). As a result, the relevant market output value is expected to increase from US$13.7 billion in 2020 to over US$32.7 billion in 2025 (https://www.marketsandmarkets.com/Market-Reports/software-defined-networking-sdn-market-655.html, accessed on 28 October 2021) with programmable network management and orchestration as critical technologies.

The software-defined network leverages the decoupling of the control and data planes to achieve centralized control of the entire system. This eliminates the need for network administrators to set routing policies independently for each network device, making the network more manageable and flexible. However, the centralized architecture of SDN renders the network susceptible to information security threats. Attackers can use various techniques to perform malicious tasks, and gaining access to the controller can put the entire network system at risk. Therefore, deploying an intrusion detection system (IDS) to detect SDN traffic is critical to this architecture.

Intrusion detection systems rely on two primary detection methods: signature and anomaly-based [1,2]. Signature-based detection requires continuous database updating and cannot identify new or unknown attacks. By contrast, anomaly-based detection establishes a benchmark for the typical behavioral characteristics of network packets and compares the data parts to be detected with this benchmark to determine abnormalities. Of these methods, anomaly-based methods are better equipped to identify attack behaviors and have played a crucial role in safeguarding networks from new attacks in recent years.

Machine-learning technology has been widely adopted in anomaly-based intrusion detection systems [3]. Depending on whether the data are labeled and the expected outcome, machine learning classifiers can be distinguished into four different approaches: (1) supervised learning, where classifiers are trained on data with pre-labeled categories to classify new data; (2) unsupervised learning, which does not rely on labeled data but instead uses a learning algorithm to analyze the feature distribution of data points to establish a classifier and classify new data; (3) semi-supervised learning, which combines the previous two methods, using a small amount of labeled data and a large amount of unlabeled data for training, first training the machine learning classifier through the labeled data, then using the partially trained algorithm to label the unlabeled data, and finally retraining the classifier using both labeled and newly labeled data; and (4) reinforcement learning, which does not use labeled data but relies on reward and punishment mechanisms and evaluation metrics to help the classifier continually try and adjust to the best parameters based on the quality of feedback.

Traditional machine learning methods assume the training and testing datasets share the same feature space and distribution. However, this assumption is not fully applicable in practice, and the classification performance may decrease if there is a gap in the sample distribution between the two datasets. Three common challenges encountered during practical scenarios are insufficient data volume, incompatibility of computing power, and distribution mismatch [4]. To address the problem of insufficient training data, transfer learning allows the classifier to be trained from related data to improve the performance of the test data classification and accelerate the learning efficiency of the target domain by transferring the learned parameters from related fields. Transfer learning enables the reuse of training knowledge from other domains and tasks, thereby reducing the time and computing resources required to train the model. Moreover, the problem of reduced classification performance caused by a distribution mismatch can be solved by synthesizing knowledge from one or more domains through various domain adaptations.

The concept of sustainability in this study includes the sustainability of SDN applications and the robust performance of transfer-learning paradigms in detecting new and unknown attacks over time. The SDN architecture enables centralized management and communication between controllers and OpenFlow switches, thereby reducing the cost of controller–switch communication for dynamic and programmatically efficient networks. For network security issues, owing to the ever-changing nature of unknown attacks and potential threats, the transfer learning model must be able to identify abnormal attacks over time while maintaining its robustness. In addition, the acquired knowledge needs to be quickly fine-tuned based on small samples and then applied to new task domains to face attacks and models requiring specific detection performance.

Transfer learning leverages the knowledge acquired from different fields to solve the target task. The classifier need not learn from scratch when new samples are added, thus solving the problem of insufficient training data. Unknown attack identification is particularly important because the number of new attack samples is usually small in the initial stage of development. However, if such attacks are tolerated or ignored, they can pose a security threat to the entire information network. Nevertheless, the traditional performance of machine-learning classification is limited by insufficient training data, highlighting the importance of addressing the insufficiency of samples.

To address the challenges of unknown attack detection, insufficient training samples, and imbalanced classification in intrusion detection systems, we propose a system architecture based on the transfer learning strategy for intrusion detection in the SDN environment. We propose a method called "Reptile-TL" that combines transfer learning (TL) and meta-learning technology. The architecture of the intrusion detection system comprises two stages: anomaly detection and attack category identification. Initially, a binary classification model is utilized for anomaly detection and assessment of packet flow as normal or abnormal. Subsequently, abnormal packet flows are passed to a multi-class classification model to identify the attack category and determine the type of attack. Experimental results demonstrate that the deep learning method performs well and maintains stability when sufficient training samples are available. However, the transfer learning methods (i.e., Reptile-TL and Convnet-TL) perform better in scenarios involving new unknown attacks and intrusion detection with small samples.

In summary, this paper makes three contributions:

- We have investigated three challenges related to intrusion detection in an SDN environment, i.e., unknown attacks, small samples, and class imbalance and conducted a comprehensive survey of relevant studies.
- We have proposed the Reptile-TL model, which employs transfer learning and meta-learning strategies to utilize the source domain to improve the performance of the target domains.
- We have compared the performance of six models for three critical issues and evaluated the advantages of transfer learning in intrusion detection based on the SDN data environment.

The remainder of this paper is organized as follows. Section 2 describes related technologies. Section 3 introduces the question definition and model architecture. Section 4 presents the experimental results and their evaluation. Finally, Section 5 summarizes the tasks and discusses future research directions.

2. Related Work

2.1. SDN and Security Issues

With recent changes in network behavior caused by the global pandemic [5], SDN has become a promising solution for network management and adaptation. It offers greater flexibility than traditional networks by utilizing SDN controllers or northbound application programming interfaces (APIs) to guide the network traffic design and communicate with the underlying hardware infrastructure through the OpenFlow protocol on the southbound interface. Unlike traditional networks that rely on dedicated architecture devices such as controllers, routers, and switches, SDN eliminates the need for such devices, resulting in improved network construction, flexible configuration, and lower deployment costs. The diversity and demand for many network function services have led to the use of virtualization technology to automate various management tasks. The use of this technology reduces human errors and difficulties faced by professionals during development, shortens deployment time and costs, and provides rapid segmentation as required by computing and storage resources [6].

Network function virtualization differs from traditional network service functions because it does not require the use of dedicated hardware equipment or underlying hardware network architecture. When adding a new system or service, in addition to considering resource requirements and service application types, the method can establish underlying network segments, routing settings, firewalls, and other traffic-related security checks, load balancing, and VPN settings required by the system according to its actual needs. Simultaneously, it can also dynamically request the corresponding composition conditions from the network virtualization system and configure the required network settings and functions. This is the concept of service function chaining. If not required, the system can be quickly recovered for subsequent demand re-planning [7].

SDN can be employed to reduce the complexity associated with the Internet-of-Things (IoT) and improve the quality-of-service (QoS). Ali et al. [8] proposed an efficient slave controller allocation-based load-balancing approach for a multi-domain SDN-enabled IoT network, which aims to effectively transfer switches to a controller with idle resources. SDN provides a practical approach to network management and adaptation by separating the control and data layers, allowing for greater flexibility in network configuration. SDN controllers or northbound APIs guide the network traffic design and communicate with the underlying hardware infrastructure on the southbound interface via the OpenFlow protocol, eliminating the need for dedicated hardware devices such as controllers, routers, and switches that are required in traditional networks. This results in improved network construction, flexible configuration, and lower deployment costs [9].

However, the centralized architecture of an SDN may be vulnerable to security threats from traditional networks and new information security vulnerabilities. Several studies [9–11] have analyzed the security threats posed to the SDN architecture. The control layer of SDN is the core component responsible for converting the priority conditions of the application program, providing precise instructions for each system component, and providing relevant information to the SDN application program. The control plane manages logically all connected OpenFlow elements and enforces the policies defined by them [11]. The controller processes devices on the data layer through channels, with each device having a separate logical channel. However, these channels share the same physical link. A controlled device can run a flood attack, causing link congestion and isolating the SDN controller from the entire network by intercepting the communication between the controller and the data layer [12]. In addition, attackers can exploit the trust between the OpenFlow switches and controllers to launch man-in-the-middle attacks to collect valuable information or gain full access to the control layer [13].

The data layer comprises network devices, such as switches, which are controlled by the control layer. These devices execute programmable control algorithms and enhance management processes and configurations for data transmission tasks. This layer requires continuous connection and monitoring of the data transfer status between end hosts. In addition, the data layer provides a standard protocol for establishing the software environment to ensure that different devices can communicate and manage the open and modular dimensions by standardizing Web applications, systems, and policies [13].

However, despite the potential benefits of SDN, unauthorized access to vulnerable hosts in the network can be exploited to launch various attacks. For example, attackers can generate malicious traffic to flood network elements with hosts or connected switches, leading to the exhaustion of the resources of OpenFlow switches' controller or flow table space. Attackers can deploy fake switches in an SDN to evade network traffic or steal target data. Moreover, they can manipulate the packet-forwarding rules (flow entry) of an OpenFlow switch to redirect legitimate network traffic, which can cause significant network congestion or disruptions. Finally, fake switches can be used to generate fake request messages to increase the load on the controller and reduce the speed of network transmission.

2.2. Machine Learning in Intrusion Detection Issues

Intrusion detection systems (IDSs) can detect attacks using signature or anomaly-based methods [1,14,15]. Signature-based detection, also known as knowledge-based or misuse detection [2], involves comparing incoming or outgoing packet traffic with a large database of known attack signatures to identify and block malicious traffic. However, this method is only effective for detecting known attacks and requires the storage of many rules. By contrast, anomaly-based detection detects deviations from normal behavior [16]. This method first establishes a baseline for normal network traffic and then compares the traffic against the baseline to detect anomalies. Anomaly-based detection can detect both known and unknown attacks. Machine learning technology is commonly used to classify network traffic as normal or abnormal and to further classify abnormal traffic to identify the type of attack.

Machine learning has been widely and successfully used in various applications, including intrusion detection. Kumar et al. [17] and Rathore et al. [18] used decision trees for intrusion detection. By constructing a tree consisting of root nodes, internal nodes, branches, and leaf nodes, internal nodes represent data features, branches represent rules, and leaf nodes represent the final labels. Determining attributes of nodes to predict data categories can help handle large amounts of data and support real-time traffic detection.

In addition, related studies have added conceived of unsupervised machine learning methods. For example, Ali et al. [19] proposed a hierarchical control plane SDN architecture for multi-domain communication using principal component analysis (PCA) to reduce the dimensionality of big data traffic and applied the support vector machine (SVM) classifier to detect a distributed denial-of-service (DDoS) attack. Lopez-Martin et al. [20] proposed a new method for detecting intrusions in IoT networks. The method incorporated intrusion labels inside the inner decoder layers based on a conditional variational autoencoder. It was less complex than other unsupervised methods. In addition, it was used to recover missing features for incomplete training datasets.

Ahmed et al. [1] noted that recent studies [21–23] have focused on using multi-layer deep neural networks to detect and classify cyber attacks, which have shown good performance in detecting network anomalies. In practical scenarios, when there is a disparity in the distribution of samples between the training and testing datasets, classification performance decreases. Inadequate training data, discrepancies in computing capacity, and differences in data distribution are commonly encountered challenges in practical situations [4]. Assis et al. [24] proposed an SDN defense system based on analyzing single IP flow records, which uses the gated recurrent unit (GRU) deep learning method to detect DDoS and intrusion attacks. Direct flow inspection enables faster mitigation responses, thereby minimizing the impact of the attack on the SDN.

2.3. Transfer Learning in Intrusion Detection Issues

The transfer-learning approach improves the performance of traditional classification models in the target domain by transferring knowledge from different source domains [25]. This strategy reduces the dependency of the classification model on the existence of a large amount of target data. Furthermore, the classification model of transfer learning has already acquired knowledge from other domains before the training starts. Therefore, it does not need to learn from scratch, unlike traditional machine-learning methods. Thus, the performance of the transfer learning classification model is usually better than that of traditional machine learning methods, and this model is more suitable for situations in which training samples are insufficient or unknown attack detection is required.

Several studies have applied transfer learning to intrusion detection to address issues such as insufficient training samples and unknown attack detection. For instance, Wu et al. [26] proposed a transfer learning-based convolutional neural network (CNN) called ConvNet-TL for network intrusion detection. The model consists of two connected CNNs, and the learning process is divided into two stages. First, a CNN model is trained on the source dataset, and the learned knowledge is then transferred to another CNN model and the target dataset. The proposed method is verified on the NSL-KDD dataset and demonstrates a better classification performance than traditional CNN models; in addition, the proposed model can improve the detection accuracy for known and unknown attacks.

Rodríguez et al. [27] proposed a CNN-TL method for detecting unknown attacks in IoT environments. The approach involved training a CNN on the BoT-IoT dataset [28], transferring its convolutional layer to a new CNN model and fine-tuning it on a traditional network dataset. The effectiveness of the transfer-learning-based IDS was tested on the UNSW-NB15 dataset [29] and the IDS was shown to effectively detect cyber-attacks in IoT circuits with small and imbalanced datasets. This approach is similar to [26]; however, this study explored the feasibility of deploying a transfer learning-based IDS to deal with unknown attacks specifically for IoT environments with limited data..

Dai et al. [30] proposed TrAdaBoost, which uses AdaBoost-based technology to select samples from the source domain that are helpful for the classification of the target domain and assign them higher weights. The model is then trained using the re-weighted samples from the source domain and a few examples from the target domain. Wang et al. [31] proposed an instance-based transfer learning method and applied it to DDoS attack detection. The method used the publicly accessible DDoS dataset as the source domain and selected the top few information gain features for the dimensional reconstruction of the dataset. The TrAdaBoost algorithm was used to determine samples that were helpful for the classification of the target domain to ensure that the model could detect unknown DDoS attack behaviors. This study focused solely on DDoS attacks and only one source domain dataset was selected with a relatively small overall data volume; if the source domain data are extensive, this approach may require a significant amount of time.

Singla et al. [32] investigated the feasibility of transfer learning in intrusion detection using UNSW-NB15 as both the source and target datasets. The target domain comprised one of the attacks in UNSW-NB15 and some normal samples, whereas the remainder belonged to the source domain. The objective was to compare the performance of intrusion-detection models which use transfer learning with that of those trained from scratch. The number of training samples in the target domain ranged from 100 to 70,000. The results indicated that transfer learning can better identify new attacks when fewer training data are available in the target domain.

Dhillon et al. [33] employed the CNN-LSTM model to implement transfer learning. UNSW-NB15 was used as both the source and target datasets, where the source domain was split into training and validation sets, accounting for 60% and 20% of the data, respectively. The testing set comprised the target domain, which accounted for the remaining 20% of the data. The CNN layer was first used to extract the features of the training dataset, down-sampling the input while retaining essential features during the extraction process to reduce the data dimension. The output of the CNN layer was then fed to the LSTM layer, and the weights were adjusted using the backpropagation algorithm. Finally, the results were passed to a fully connected layer for classification. The method achieves classification accuracies of 98.30% and 98.43% for the source and target datasets, respectively.

Dos Santos et al. [34] proposed a deep autoencoder and transfer learning-based intrusion detection model to alleviate the model update burden in real networks. The model leverages a deep auto-encoder as an additional feature extraction stage to obtain a historical feature representation of network traffic. When the model is updated, some parameters of the deep autoencoder are retained through transfer learning, and the classification model is trained using new samples to update the parameters. This approach significantly reduces the amount of labeled training data and the computing costs required to effectively distinguishing between unusual and normal wireless network traffic.

Sameera et al. [35] utilized the manifold alignment transfer learning method to transform the source and target domains into a common latent space, circumventing the problem of different marginal probability distributions between the feature spaces and domains. They proposed a method for generating labels in the transformed space, utilizing a clustering correspondence procedure to compensate for the lack of labeled object samples. The classification model uses the DNN framework and is evaluated on the NSL-KDD [36] and CIDD datasets [35]. The experimental results showed that the proposed method can successfully detect some unknown attacks, even if the distributions of the source and target domains differ; however, the F1-score is only 0.4.

Phan et al. [37] proposed a CNN-based transfer learning model called DeepFlowIDS. The model first extracts important features and is trained on the NSL-KDD dataset. The abnormal detection performance of the model is then evaluated for the DDoS attack category of CIC-IDS2018. Finally, the model is tested in a simulated SDN environment. However, only one type of DDoS attack is assumed for an unknown attack; therefore, the diversity of attack types is limited.

Similarly, Polat et al. [38] proposed a transfer learning model based on time series to detect a type of DDoS attack on a supervisory control and data acquisition (SCADA) system in a simulated SDN environment. Three types of DDoS attacks were considered, and the model was trained and tested on a dataset of normal network traffic. This method utilizes two independent deep learning models: LSTM and GRU. After training on 90% of the dataset, the parts other than the output layer of the two models are extracted as feature extractors of the transfer model. The extracted features are then processed using an SVM for multi-class classification.

Polat et al. employed the Q-learning [39] , a reinforcement learning technique, to acquire the most informative knowledge from the source domain for the target task and provide an optimal initial model parameter for the MLP deep learning model to enhance the performance of the target domain. The proposed method trains the model using four datasets from conventional network environments: NSL-KDD, UNSW-NB15, CICIDS, and CICDDoS. It applies the model for DDoS anomaly detection in an SDN environment. The classification accuracy of the proposed method outperforms other transfer learning methods that do not utilize reinforcement learning techniques.

The research cited above is listed in Table 1. The objective of these studies was to distinguish between normal and abnormal network traffic without employing multi-class classification techniques to identify different types of abnormal traffic. However, we believe that the intrusion detection systems can be refined and automated by classifying abnormal traffic into various types of attacks. This would allow system administrators to implement targeted responses according to the type of attack, thereby significantly reducing the time required for manual interpretation.

Furthermore, while many existing studies address common issues such as limited training samples or unknown attack detection, they tend to focus on one problem rather than comprehensively evaluating all three issues. In addition, most of the current research, such as in [26,31–33,40,41], has been conducted in traditional networks. In contrast, a few others, including [27,34,35], have explored the IoT, wireless networks, and cloud networks. This study adopts a transfer learning approach for intrusion detection in SDN environments, as described in [38,39,42], with a specific focus on DDoS attacks. However, the intrusion detection performance of this model has not been compared with that for other types of unknown attack behaviors.

Table 1. Research related to intrusion detection. (Symbol ○ indicates it belongs to.)

Author	Model	Data	Environment			Problem			# of Class	
			TN [1]	IoT	SDN	UA [2]	SS [3]	CI [4]	Binary	Multiple
Our study	CNN,MAML, Meta-SGD, Reptile, Convnet-TL, Reptile-TL	CIC-IDS InSDN			○	○	○	○	○	○
[27], 2022	CNN, Convnet-TL	UNSW-NB15 NSL-KDD	○			○			○	
[40], 2022	CNN-TL, CNN,IDS	BoT-IoT UNSW-NB15	○			○		○	○	
[38], 2022	LSTM,GTU SVM	simulation			○					○
[43], 2021	CNN,TANN, EDM,MCA, ODM	UNSW-NB15, Bot-IoT	○	○			○		○	
[44], 2021	CNN	NSL-KDD	○		○	○			○	
[45], 2021	ID-FSCIL, MAML,NB, Logistic, DT	NSL-KDD	○				○		○	
[31], 2021	TrAdaBoost	NSL-KDD, DARPA2000, CICIDS2017, UNSW-NB15	○			○			○	

Table 1. *Cont.*

Author	Model	Data	Environment			Problem			# of Class	
			TN [1]	IoT	SDN	UA [2]	SS [3]	CI [4]	Binary	Multiple
[34], 2021	DeepAE, DT,GBT	MAWIFlow	○				○		○	
[31], 2021	SVM,KNN, CNN	KDD99, NSL-KDD	○				○		○	
[37], 2021	MLP,RL	NSL-KDD, UNSW-NB15, CICIDDoS, DosinSDN			○	○			○	
[46], 2020	FC-Net	ISCX2012, CIC-IDS2017	○			○	○		○	
[47], 2020	J48,NB,RF, SVM,RNN, DNN CBR-CNN	NSL-KDD, UNSW-NB15	○				○	○	○	○
[33], 2020	CNN-LSTM, CNN,DNN	UNSW-NB15	○				○		○	
[35], 2020	DNN,KNN, SVM,RF,DT	NSL-KDD CIDD	○	○			○		○	
[48], 2020	CNN	UNSW-NB15	○				○	○	○	
[26], 2019	CNN, Convnet-TL	UNSW-NB15 NSLKDD	○			○			○	
[32], 2019	DNN	UNSW-NB15	○				○		○	
[41], 2018	SVM,NAMA	KDD99, KYOTO2006	○			○			○	

[1] TN: Traditional Network. [2] UA: Unknown Attacks. [3] SS: Small Samples. [4] CI: Class Imbalance.

3. Methodology

3.1. Question Definition

Supervised learning methods generally assume that the training and testing data are drawn from the same distribution, such that the input feature space and data distribution remain constant [4]. However, novel forms of attacks frequently emerge in real-world networks. In the early stages of such attacks, the number of available samples is typically limited; directly using such a small sample size to train a classifier often leads to overfitting. Therefore, transfer learning leverages data from related domains to train classifiers and alleviates the problem of insufficient training data, thereby improving the classifier performance for a limited number of categories.

We propose a novel approach called Reptile-TL, which combines transfer learning and meta-learning techniques to address three critical intrusion detection challenges—unknown attacks, small sample sizes, and class imbalance—to identify abnormal detection and attack types. In transfer learning, the target domain D^T serves as the small training and testing datasets for classification tasks. The source domain dataset D^S is utilized to improve the classification performance of the target domain and is referred to as the source domain. Each domain consists of the feature space X and the marginal distribution $P(X)$ of the feature space, such that $D = \{X, P(X)\}$, where $X = \{x_i \in X, i = 1, \ldots, \text{n}\}$. Each domain is associated with a specific classification task (T), which includes a label space Y and a decision function F, such that $T = \{Y, F\}$, and F is learned from training samples to make predictions. In this study, we aim to address the following three classification problems: (1) Unknown attacks: This refers to an environment with unknown attacks where the source and target domains belong to different datasets ($D^S \neq D^T$), in which the tasks of the two domains' are related. In this scenario, some attacks on the target domain D^T do not occur in the source domain D^S. (2) Small samples: This describes a small sample environment where there are limited training samples D^T for most attack categories. (3) Class imbalance: This pertains to a class-imbalance environment, where D^T is a highly imbalanced dataset.

Our intrusion detection method is divided into two stages: anomaly detection and attack category classification, as shown in Figure 1. First, anomaly detection uses a binary classification model to determine whether a packet is normal or abnormal. If it is abnormal, it is sent to a multi-classification model to identify and classify the type of attack. This enables system administrators to perform further defense depending on the attack type. The seven attack types are determined by the multi-class classifier in the second stage: DDoS, Probe, DoS, BFA, Web attack, Botnet, and U2R.

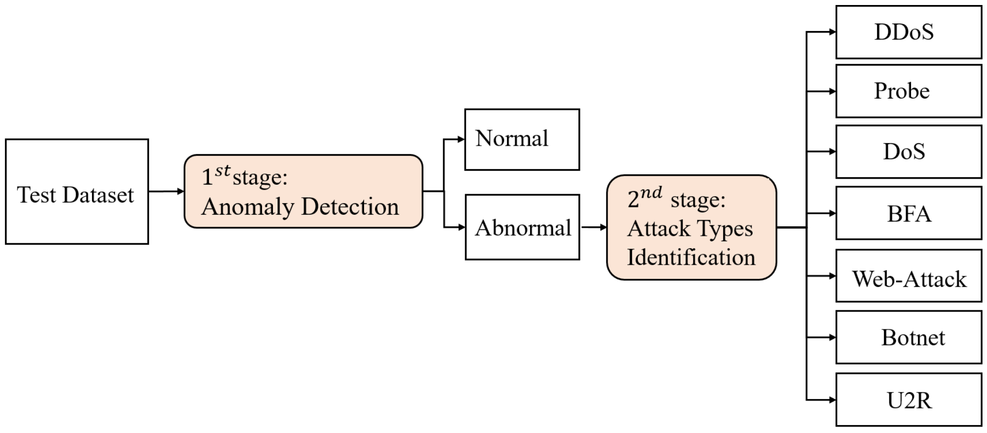
Test Dataset
1st stage: Anomaly Detection
Normal
Abnormal
2nd stage: Attack Types Identification
DDoS
Probe
DoS
BFA
Web-Attack
Botnet
U2R

Figure 1. Proposed intrusion detection architecture process.

3.2. Model Architectures

The conventional supervised learning approach for intrusion detection requires a priori knowledge of all attack categories and adequate training samples for each type. However, new attacks emerge almost daily, and their samples may not be available beforehand. As a result, the number of samples is often limited, leading to poor performance or overfitting. In such scenarios, traditional supervised learning methods may be unable to make effective predictions because of inadequate training.

Meta-learning has been proven to be an effective solution for small-sample classification problems. In meta-learning, each meta-training stage comprises multi-class classification tasks, where each task consists of a support set and query set both sampled from the same training dataset. The model is trained on the support set and validated on the query set; this process is repeated several times. The aim is to learn from a few known samples in the meta-training stage, obtain an effective feature representation, and quickly adapt to various new tasks.

1. CNN

   The CNN architecture is used to detect unknown attacks in traditional networks and has demonstrated excellent binary classification performance by Das et al. [40]. Therefore, we use the model as a basic component for binary and multi-class classification tasks in the SDN environment. The CNN model adopted for binary classification in this study composes five hidden layers; its architecture and parameters are illustrated in Figure 2. Binary cross-entropy serves as the loss function, and Adam is used as the optimizer. The output layer has two output nodes with the sigmoid activation function. The architecture of the multi-class classification model is the same as that of the binary classification model, except for the output layer, which consists of seven output nodes for the softmax activation function. Categorical cross-entropy is used as the loss function.

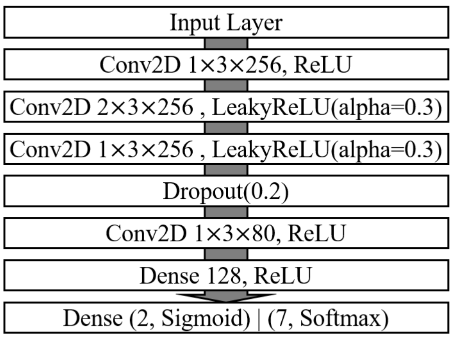
Input Layer
Conv2D 1×3×256, ReLU
Conv2D 2×3×256 , LeakyReLU(alpha=0.3)
Conv2D 1×3×256 , LeakyReLU(alpha=0.3)
Dropout(0.2)
Conv2D 1×3×80, ReLU
Dense 128, ReLU
Dense (2, Sigmoid) | (7, Softmax)

Figure 2. The CNN model for binary and multi-class classification.

2. Model-Agnostic Meta-Learning(MAML)

The MAML model aims to find model parameters that are sensitive to changes in tasks, so that good classification results can be achieved with only a small number of gradient descents and a small amount of training data. This basic model is responsible for learning the features of each task, and a meta-learning module is subsequently used for learning. The meta-learner updates the parameters of the basic model to obtain a good initial value for performing new classification tasks. The meta-learner synthesizes the training experience of all tasks, and passes the calculated initial values of the model parameters to the basic model to perform well after a few iterations. During the training process, a small number of samples are randomly selected from the labeled training dataset (support set) to perform a training task, after which the model is trained. The MAML process is illustrated in Figure 3 and proceeds as follows.

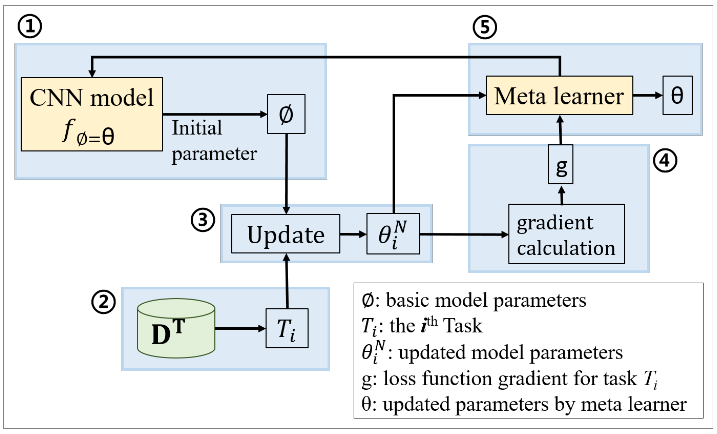
①
CNN model
$f_{\emptyset=\theta}$
Initial parameter
$\emptyset$
⑤
Meta learner
θ
④
g
gradient calculation
③
Update
θ_i^N
②
D^T
T_i
$\emptyset$: basic model parameters
T_i: the ith Task
θ_i^N: updated model parameters
g: loss function gradient for task T_i
θ: updated parameters by meta learner

Figure 3. MAML training process.

(1) The basic model f_ϕ is defined, where ϕ is a trainable parameter.
(2) Task T_i is randomly extracted from the training dataset D^T according to the probability distribution $p(T)$. The loss function of the basic model for task T_i is denoted as $L_{T_i}(f_\phi)$, and the objective function minimizes $L_{T_i}(f_\phi)$.
(3) The parameter θ is provided by the meta-learner and is initially set to ϕ. It is then updated iteratively using the loss function gradient on task T_i. After N iterations, the parameters are denoted as θ_i^N. The value of N depends on the number of samples in task T_i.
(4) After N iterations, the meta-learner returns the updated parameter θ_i^N and the gradient value of the loss function, denoted by Equation (1), to the basic model.

$$g = [\nabla_\phi L_{T_i}(f_\phi)]\phi = \theta_i^N \tag{1}$$

(5) The meta-objective function is then defined as $min \sum_{T_i \backsim P(T)} L_{T_i}(f_{\theta_i}^N)$, where $L_{T_i}(f_{\theta_i}^N)$ is the loss function based on the parameter θ_i^N on task T_i. The meta-objective function is the sum of the loss function values of the query set for all tasks. Finally, the updated initial parameter θ is obtained as shown in Equation (2), where β is the learning rate of the meta-learner, and $\bigtriangledown_\theta L_{T_i}(f_\phi)_{\phi=\theta_i^N}$ is the gradient of the loss function value after iterative training on task T_i to the initial value of the parameter.

$$\theta \leftarrow \theta - \beta \bigtriangledown_\theta \sum_{T_i \backsim p(T)} L_{T_i(f_\phi)_{\phi=\theta_i^N}} \tag{2}$$

when a new task is encountered, the meta-learner provides the updated initial parameter values for the basic model of the new task. In this study, the CNN model, which is responsible for learning the features of each task, is adopted as the basic model for intrusion detection. The meta-learning module is used to update the model parameters of the basic model for new classification tasks, thus achieving good performance with only few-shot supervised learning.

3. Meta-SGD

Meta-SGD is a variation of MAML that modifies the learning process, as illustrated in Figure 4. Unlike MAML, Meta-SGD requires updating the initial parameter θ and the corresponding learning rate of each task because the learning rate is not fixed. This necessitates the dynamic updating of the learning rate after each training iteration, resulting in high training complexity. However, all other calculations are the same as those in MAML. The objective of the meta-learner is to identify the optimal initialization parameters and learning rate vectors and provide feedback to the basic model. The model architecture is identical to that of MAML. The training parameters are represented by Equation (3), where the Nth iteration and the initial parameter update of the meta-learner are expressed by Equation (4):

$$\theta_i^N = \theta_i^{N-1} - \alpha[\bigtriangledown_\phi L_{T_i}(f_\phi)]\phi = \theta_i^{N-1} \tag{3}$$

$$(\theta, \alpha) = (\theta, \alpha) - \beta \sum T_i \sim p(T) \bigtriangledown_\theta [L_{T_i}(f_\phi)]_{\phi=\theta_i^N} \tag{4}$$

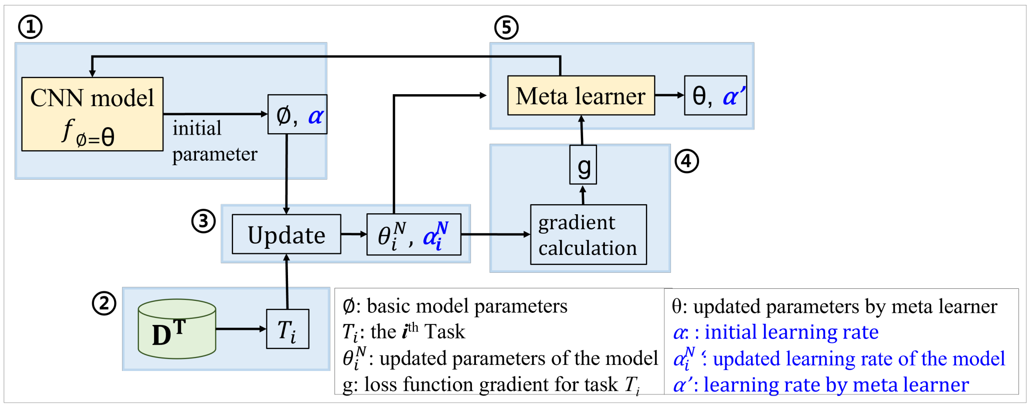
①
⑤
CNN model
$f_{\emptyset=\theta}$
initial parameter
∅, α
Meta learner
θ, α'
g
④
③
Update
θ_i^N, α_i^N
gradient calculation
②
D^T
T_i
∅: basic model parameters
T_i: the ith Task
θ_i^N: updated parameters of the model
g: loss function gradient for task T_i
θ: updated parameters by meta learner
α: initial learning rate
α_i^N: updated learning rate of the model
α': learning rate by meta learner

Figure 4. Meta-SGD training process.

4. Reptile

Similar to MAML, the Reptile method performs model-agnostic meta-learning to learn and perform new tasks quickly with minimal training data. By leveraging the difference between the weights trained on the task and the model weights before training to perform a stochastic gradient descent, the model can be rapidly generalized to a small number of samples, thereby improving its performance. Figure 5 illustrates the training process of the Reptile method.

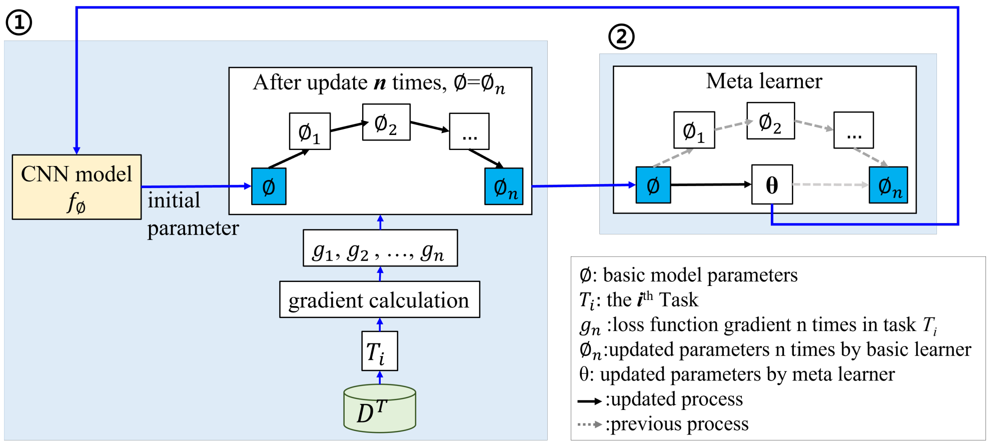
①
②
After update n times, ∅=∅_n
CNN model f_∅
initial parameter
∅
∅_1
∅_2
...
∅_n
g_1, g_2, …, g_n
gradient calculation
T_i
D^T
Meta learner
θ
∅: basic model parameters
T_i: the i^th Task
g_n :loss function gradient n times in task T_i
∅_n:updated parameters n times by basic learner
θ: updated parameters by meta learner
:updated process
:previous process

Figure 5. Reptile training process.

(1) Each task begins with the initial values of the basic model parameters, and the gradient is updated N times. The resulting estimated parameter value is then fed back to the meta-learner. In this experiment, the basic model is a CNN with the same architecture as that of MAML.

(2) The meta-learner calculates the difference vector between the previous parameter and the initial parameter and updates the initial value using Equation (5), where ϵ is the learning rate, N is the number of iterative updates of the initial value of the parameters on the basic learner, and n denotes that the initial value of the parameters in the meta-learner is updated once after training the basic model on n tasks each time.

$$\theta \leftarrow \theta + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\theta_i^N - \theta) \tag{5}$$

5. ConvNet-TL

The ConvNet-TL approach was first proposed by Wu et al. [26]. This method aims to enhance the detection performance for unknown attack anomalies in traditional network datasets by transferring the knowledge learned from these datasets. Building on this idea, our study applies this method to anomaly detection and multiclassification in an SDN environment and investigates whether transferring knowledge from traditional network datasets can enhance the classification performance of models in an SDN environment.

6. Reptile-TL

Because conventional supervised learning relies heavily on several training samples, training models with small samples can result in overfitting or underfitting. Thus, we propose the Reptile-TL model, which combines the strengths of traditional deep learning and meta-learning. Specifically, we use traditional deep learning methods to train models on the source domain with many samples, effectively learning and transferring knowledge. However, for the target domain with a small number of samples, we use the Reptile method to train the model and improve classification performance. Our model for the training process comprises two stages, as illustrated in Figure 6. The first stage involves the basic model training, in which a CNN model is trained in the source domain D^S. The CNN architecture for binary and multi-class classification is shown in Figure 7, with its four convolutional layers serving as feature extractors to extract important features of the source domain to aid the training in the second stage. The purpose is to train the Convnet_B model in D^S with sufficient samples and then migrate its convolutional layer to the Reptile-TL model such that Reptile-TL can be trained in the target field and have good initial parameters. When preparing the migration model in the second stage, Reptile-TL updates the initial parameters of the model using Reptile meta-learning to obtain better initial values of model parameters. During training, the parameters of the convolutional layer model transferred from ConvNet_B are fixed to avoid overfitting. The loss function of the binary classification model uses binary cross-entropy, whereas that of the multiclass classification model uses categorical cross-entropy, and the optimizer is Adam.

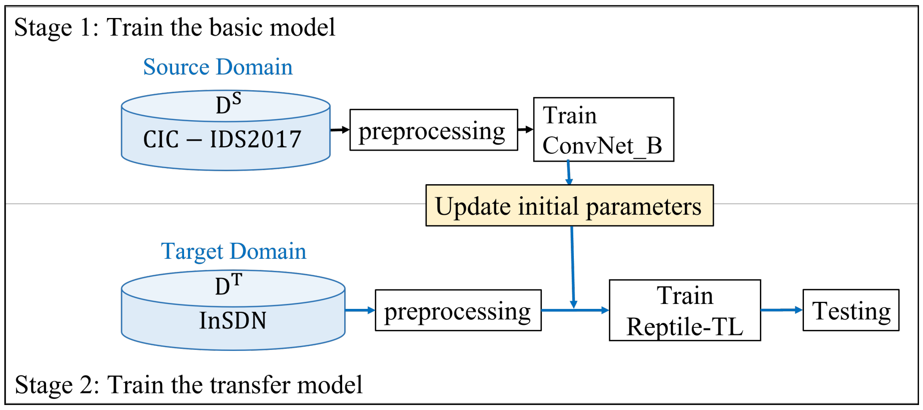

Figure 6. Proposed transfer classifiers processes.

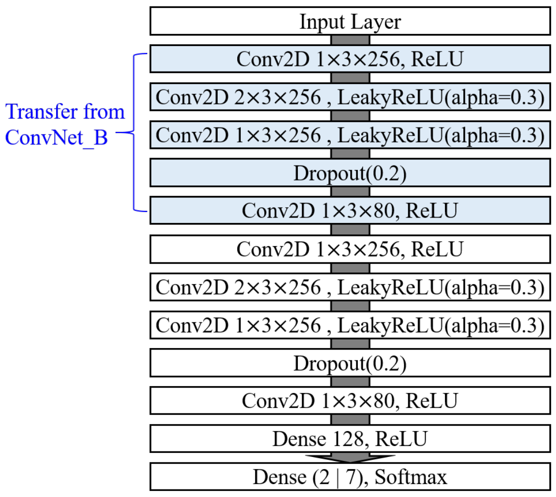

Figure 7. Reptile-TL architecture.

4. Experiments

4.1. Datasets and Measures

We investigated the three discussed issues with the intrusion detection framework in SDN environments. Our model utilized two datasets to demonstrate the performance of transfer learning: one for the source domain and the other for the target domain, as listed in Table 2. For the source domain, we used the CIC-IDS2017 dataset [49] developed by the Canadian Institute for Cybersecurity (CIC), which contains benign and various attack types observed in traditional networks. For the target domain, we used the InSDN dataset proposed by Elsayed et al. [12], which collected normal and seven types of attacks in SDN and was currently the latest intrusion detection dataset based on the SDN environment. We relabeled the labels of attack types as 0 and 1 from the source and target domain datasets, respectively, used the source dataset to train the parameters of the source-domain model, and then applied part of the target dataset for training to improve performance and efficiency.

Table 2. Data labeling of datasets and attack types.

Label	**Source Domain CIC-IDS2017** [49]	**Target Domain InSDN** [12]
0	Benign	Normal
1	DoS, DoS goldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris, FTP-Patator, Heartbleed, Infiltration, PortScan, SSH-Patator, Web attack, Brute Force, SQL Injection, XSS	DDoS, DoS, Probe, BFA, Web attack, Botnet, U2R

The CIC-IDS2017 dataset is a traditional network-based intrusion detection dataset comprising 14 attack types and normal traffic data. This dataset was developed in 2017 by the Canadian Institute of Cyber Security at the University of New Brunswick (UNB) [49] and contains 78 features. The dataset comprises 80% normal traffic and 20% attack traffic. The InSDN dataset contains seven types of attacks: DDoS, DoS, prospecting, botnet, brute force attack, Web attack, and User to Root (U2R). It contains 83 features, with 19.9% normal traffic and 80.1% abnormal traffic.

To prepare the dataset for unknown attacks, only DDoS attacks were retained in the training dataset. By contrast, the testing dataset included six other attack types as unseen types: Probe, DoS, BFA, Web attack, Botnet, and U2R. The numbers of the various attack types in the training and testing datasets are listed in Table 3. Next, to compare the performance based on few-shot learning, we prepared the dataset by adding 5, 10, and 15 samples of the attack types to the original training dataset to generate three sample sets—5-shot, 10-shot, and 15-shot, respectively—to evaluate the learning curve of few-shot learning. Finally, we prepared the imbalanced dataset, as shown in Table 4. The dataset was randomly sampled and divided into 75% training and 25% testing datasets with a total of one normal and seven abnormal attack types.

Table 3. Data preparation of target domain (InSDN) for unknown attacks.

	Types	**# of Training**	**# of Testing**
Known	Normal	34,212	34,212
	DDoS	121,942	0
Unknown	Probe	0	98,129
	DoS	0	53,616
	BFA	0	1405
	Web attack	0	192
	Botnet	0	164
	U2R	0	17

Table 4. Data preparation for class imbalance.

Class	**# of Training (75%)**	**# of Testing (25%)**
Normal	51,292	17,132
DDoS	91,354	30,588
Probe	73,696	24,433
DoS	40,206	13,410
BFA	1079	326
Web attack	144	48
Botnet	132	32
U2R	14	3

We used the F1-score as an evaluation metric to balance the precision and recall, which is more objective than accuracy, particularly when the data distribution is imbalanced. The equation is shown in Equations (6)–(8), where True Positive (TP) and True Negative (TN) represent normal and the number of correct predictions for the attack type, while False Negative (FN) and False Positive (FP) represent the number of misclassified predictions for normal and attack types.

$$\text{F1-score} = \frac{2P \times R}{P + R} \tag{6}$$

$$P = \frac{TP}{TP + FP} \tag{7}$$

$$R = \frac{TP}{TP + FN} \tag{8}$$

4.2. Performance Comparison in Three Issues

In the experimental design, we compared the basic deep-learning model (CNN), four types of meta-learning models (MAML, Meta-SGD, Reptile, and ConvNet-TL), and our model (Reptile-TL). We conducted this experiment on the 12th generation Intel(R) Core(TM) i7-12700H 2.70 GHz CPU, and each model was implemented using Python 3.9.13, TensorFlow version 2.11.0, and Keras 2.11.0. The simulated datasets are publicly available.

4.2.1. Unknown Attacks

The first experiment was used to evaluate the anomaly detection performance of the six models for unknown attacks, as shown in Figure 8. The results showed that Reptile-TL had the highest F1-score with an average of 0.71, which was followed by Reptile (0.65), and Convnet-TL method (0.52). We found that the performances of the two models based on transfer-learning (Reptile-TL and Convnet-TL) were higher than that of CNN (0.34). As shown in Figure 9, we compared the confusion matrices of Reptile and Reptile-TL for error analysis; we set positive as abnormal and negative as normal. From the confusion matrix on the right, TP i.e., the correct detection rate (45.33%) of the anomaly category and is higher than the detection rate (39.71%) on the left. Therefore, Reptile-TL is significantly better than Reptile. Further analysis of the error types shows that the proportion of Reptile-TL results in FN was more significant than that in FP (36.45% > 0.23%). The model misjudged unknown attacks as normal packets more and misjudged normal packets as abnormal less. Thus, future studies should use weighted adjustment parameters.

We believe that deep learning relies on a large amount of labeled data for high-performance training. The anomaly detection performance was lower for various types of unknown attacks. In contrast, the transfer learning models (Reptile-TL and Convnet-TL) learned from the source domain before training the target domain. Although the source domain is a traditional network environment, unlike the SDN environment of the target domain, the source domain enhances the performance of the models in performing classification tasks in the target domain (unknown attacks). Additionally, the Reptile, MAML, and Meta-SGD models learned suitable initial parameters from the source domain during training. Therefore, in the face of unknown attacks, the performance of anomaly detection is often better than that of a basic deep learning model (CNN).

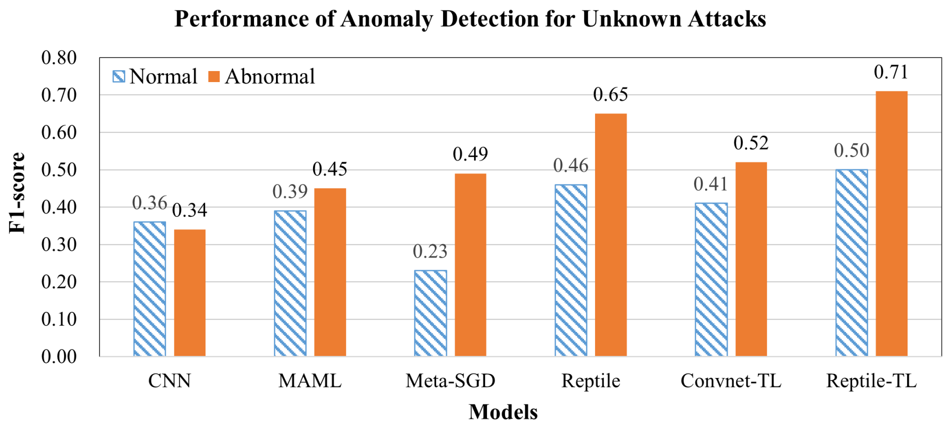

Figure 8. Performance of anomaly detection for unknown attacks.

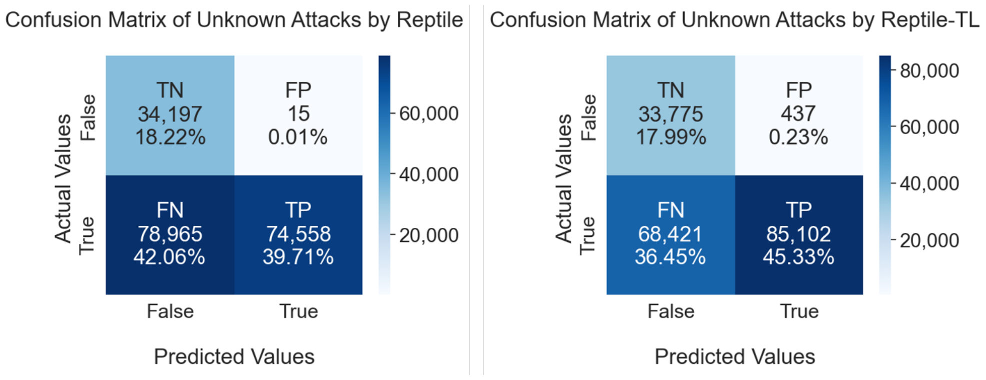

Figure 9. Confusion matrix of unknown attacks by Reptile and Reptile-TL.

4.2.2. Small Samples

We compared three types of small samples: 5-shot, 10-shot, and 15-shot datasets to show the learning curve of few-shot learning. Among the six models, Reptile-TL had the best average F1-score for anomaly detection, with an F1-score of 0.98, which was followed by Reptile (0.92) and Convnet -TL (0.85), as shown in Figure 10. Reptile-TL combined the advantages of Reptile for small samples; therefore, its sensitivity is higher than that of ConvNet-TL. The results of the two transfer learning models (Reptile-TL and Convnet-TL) were better than those of the CNN model (0.63), again indicating that the learning knowledge transferred from the source domain was critical when target domain data were rare. In contrast, the CNN model relied on many samples for training; therefore it could not effectively extract important features from small samples for attack categories with few examples, so the performance was poor. As shown in Figure 11, we compared the confusion matrices between Reptile and Reptile-TL, setting positive as abnormal and negative as normal. The confusion matrices show that the TP of Reptile-TL was higher than that of Reptile (66.31% > 61.90%); therefore Reptile-TL was significantly better than Reptile. Further analysis of the error types showed that the proportion of Reptile-TL results in FN was 15.46%, and the FP error rate was 0.02%.

Next, we compared the performance of attack-type identification in the second stage, as shown in Figure 12. Because the first stage was only used as an intrusion detection system for normal or abnormal detection, the binary classification model had high performance and avoided affecting normal packet delivery.

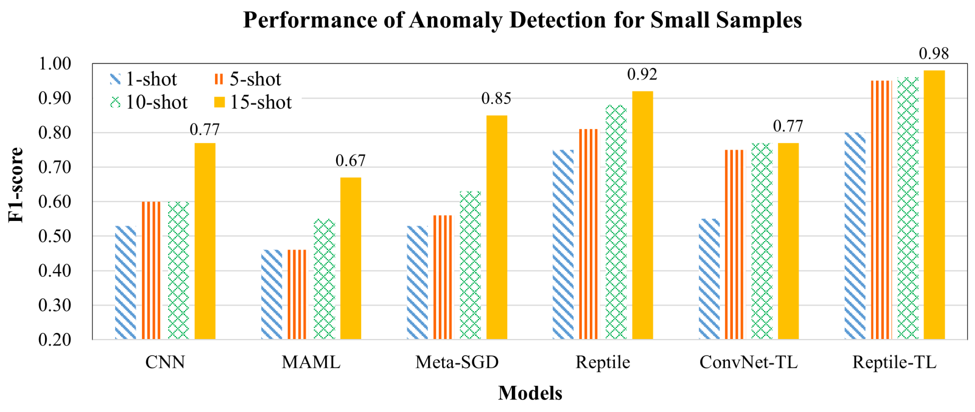

Figure 10. Performance of anomaly detection for small samples.

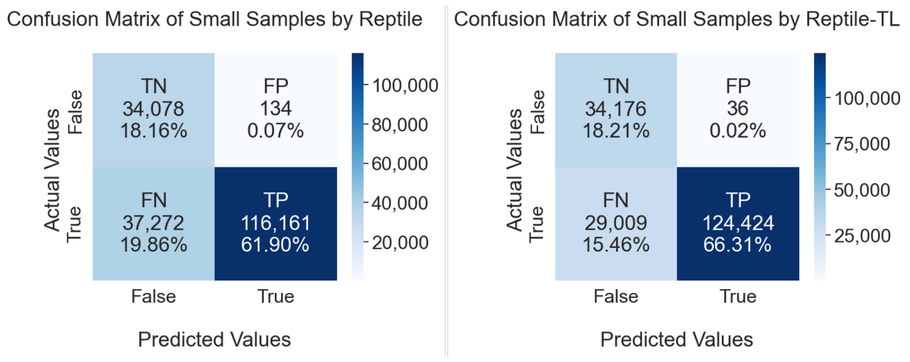

Figure 11. Confusion matrix of small samples for Reptile and Reptile-TL.

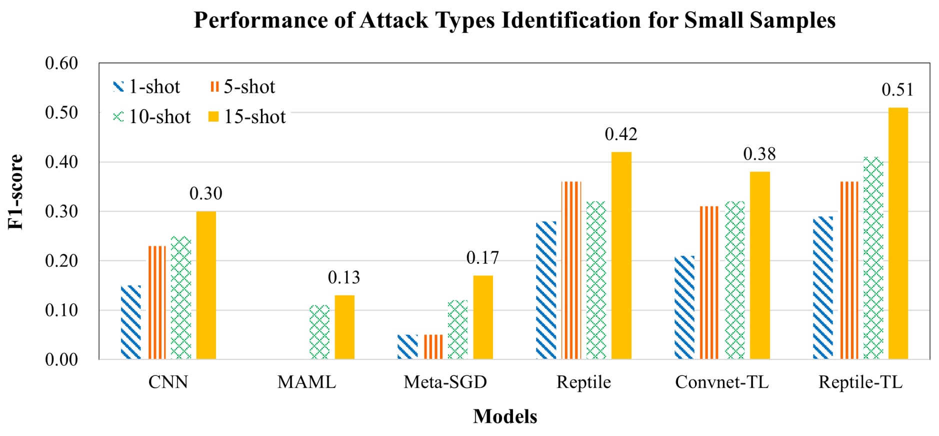

Figure 12. Performance of attack types identification for small samples.

The model of the second stage, which was used to predict the labels of the attack type identification, provided system administrators with further follow-up actions. However, because there were seven types of attacks, the number of training samples was small, and the performance of each class was easily affected by the imbalanced dataset. Thus, the performance of the six models was generally slightly poor. However, our Reptile-TL model ranked first in the performance of the six models (0.51 F1-score), which showed that the advantages of transfer learning achieved the performance of multi-class classification in small samples.

As shown in Figure 13, we compared the confusion matrices of ConvNet-TL and Reptile-TL. We set 0 to 7 as eight types: normal, DDoS, Probe, DoS, BFA, Web attack, Botnet, and U2R. For the six types of attacks, the TP of Reptile-TL was higher than that of Reptile; therefore, Reptile-TL was slightly better than Reptile. Further analysis of the error types showed that the number of FPs was greater than that of FN for Reptile-TL. Some DDoS attacks were misjudged as DoS attacks, while some probes were misjudged as normal. These results indicate that the features of the two attacks may be similar, and future research should further strengthen the weights of the differences between these two attack types.

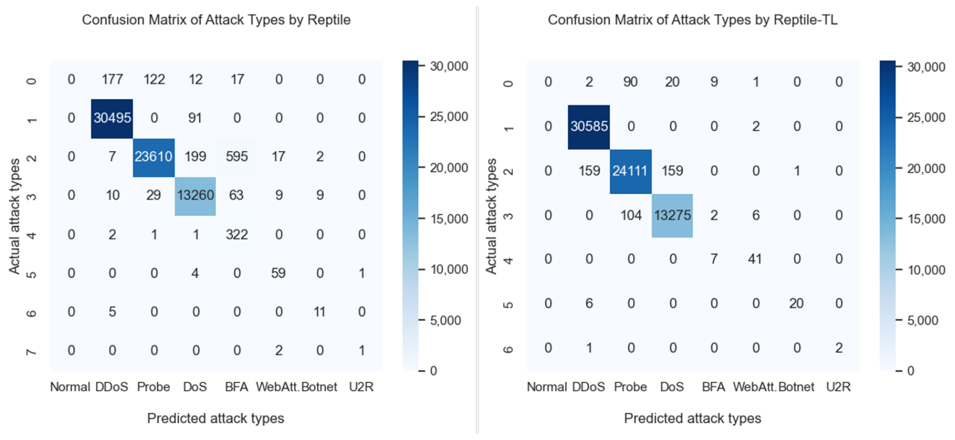

Figure 13. Confusion matrix of attack types identification for Reptile and Reptile-TL.

4.2.3. Class Imbalance

The third experiment was conducted to evaluate the performance of the six models' for class imbalance in anomaly detection, as shown in Figure 14. Because the number of training samples was sufficient to train the models to determine the binary classification (i.e., abnormal or normal), the F1-score of the models was as high as 1.00 for CNN, Reptile, ConvNet-TL, and Reptile-TL. As shown in Figure 15, we compared the confusion matrices of ConvNet-TL and Reptile-TL, setting positive as abnormal and negative as normal. The confusion matrices indicate that the TP of Reptile-TL was higher than that of Reptile (66.31% > 61.90%); therefore Reptile-TL performed significantly better than Reptile. Further analysis of the error types showed that the proportion of Reptile-TL results in FN was 15.46%, and the FP error rate was 0.02%.

Next, for attack-type identification, the experimental results demonstrated the advantages of transfer learning, as shown in Table 5. Although the performances of Reptile-TL and CNN were roughly equal, with F1-scores of 0.91 and 0.90, respectively, Convnet-TL is third, with an F1-score of 0.85. The performances of the minority classes (i.e., U2R, Botnet, and Web attack) of Reptile-TL obtained better performance than the other models. Regarding multi-class classification, the typical models were easily affected by the number of majority classes when the imbalanced dataset, especially since the number of U2R in the training sample was only 14. Therefore, the performance of the minority classes remains critical for attack-type identification.

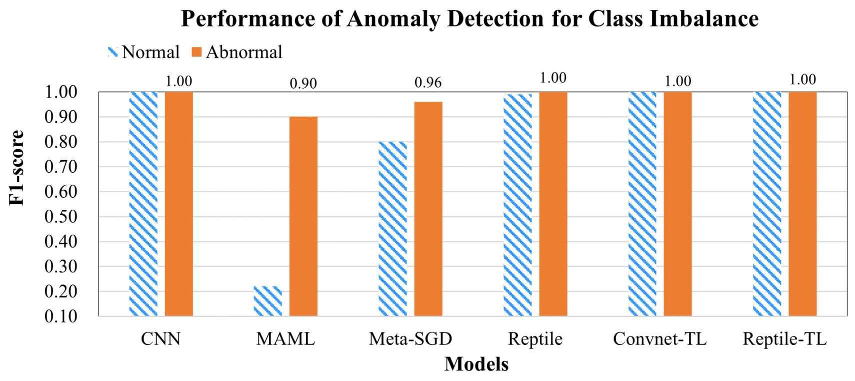

Figure 14. Performance of anomaly detection for class imbalance.

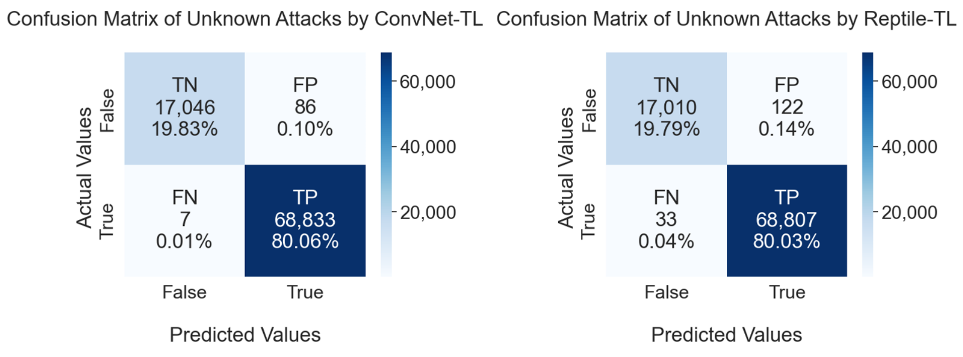

Figure 15. Confusion matrix of class imbalance by ConvNet-TL and Reptile-TL.

Table 5. Performance for class imbalance with seven types of attacks.

Class	CNN	MAML	Meta-SGD	Reptile	ConvNet-TL	Reptile-TL
DDoS	1.00	0.99	0.98	1.00	1.00	1.00
Probe	0.99	0.21	0.46	0.98	1.00	1.00
DoS	1.00	0.56	0.61	0.98	1.00	1.00
BFA	0.91	0.03	0.00	0.98	1.00	0.99
Web attack	0.83	0.00	0.00	0.78	0.43	0.84
Botnet	1.00	0.04	0.01	0.58	0.99	0.85
U2R	0.60	0.00	0.00	0.40	0.67	0.80
macro-F1	0.90	0.26	0.29	0.74	0.85	0.91

The results of the multiple attacks are listed in Table 5. For the majority of classes (i.e., attack types with a large number of samples, DDoS, Probe, and DoS), the CNN, Reptile, ConvNet-TL, and Reptile-TL models achieved good performance (larger than 0.98). Because the core model of Reptile-TL is CNN, the performance of Reptile-TL was similar to that of CNN. However, the number of training samples in the target domain was sufficient, and the performance of the transfer-learning models was not significant. When the training samples were small or insufficient, the advantages of the transfer learning models were effectively applied. The results demonstrate that the performance of our Reptile-TL model was better than that of the other models.

Additionally, correctly classifying minority classes in an imbalanced dataset is challenging—a critical problem in typical multi-class classification models. MAML exhibited the lowest performance in this experiment, possibly because the task involved multi-class attacks. The difference between each attack was significant, which rendered the model unable to generalize well; thus, its performance was poor. Because Meta-SGD is a modified version of MAML, the learning rate was adjusted for each training; thus, its performance was better than that of MAML.

Finally, the training times of the six models were compared using an imbalanced dataset; the results are shown in Table 6. The most efficient model with the least training time was Reptile, which required 14,109 s (3:55′:9″). The proposed Reptile-TL ranked second among the six models, with a training time of 15,230 s (4:13′50″). In contrast, although CNN ranked second for attack type identification performance in class-imbalance issues among the six models, it took the longest training time, with 30,088 s (8:21′:28″). Therefore, our model not only has the characteristics of transfer learning but also considers the anomaly detection performance and efficiency required for the actual training model.

Table 6. Efficiency comparison of six models during training and testing stages.

Time (Second)	CNN	MAML	Meta-SGD	Reptile	ConvNet-TL	Reptile-TL
Training	30,088	21,518	25,092	14,109	28,924	15,230
Testing	10,539	7053	8288	6416	16,659	9403

4.3. Discussion

In this subsection, we explain the feature differences between SDN and traditional networks, illustrate the detection differences between active and passive attacks, and presenting the challenges of the proposed method from other perspectives.

4.3.1. Feature Differences of between SDN and Traditional Networks

Regarding intrusion detection, the differences between SDN and traditional networks mainly include traffic characteristics and attack behaviors, as described below.

Traffic features. In traditional networks, traffic features are primarily collected from various network devices; whereas in SDN, the traffic information of SDN controllers is directly obtained from switches using the OpenFlow protocol. For example, the InSDN dataset [12] collected from the controller includes the duration of the flow, type of transmission protocol, the maximum timeout time of the flow, idle time of the flow, number of packets transmitted and received from the source to the destination, number of packets and packet size for simultaneous bidirectional data transmission in the network, and standards for the subsequent manual calculation of traffic features. There are 48 features, including difference, minimum value, maximum value, and mean value. Although some of the names of these features collected in the SDN environment are the same as those in the traditional network dataset, considering the CIC-IDS2017 dataset [49], there are 73 common features between the two. However, SDN is more sensitive to attacks owing to different network architectures. Therefore, what is considered normal traffic in traditional networks may be regarded as attack behavior in the SDN environment. Therefore, by collecting the features of packet traffic in SDN, we can analyze the behavior of network traffic, which to detect abnormal traffic and identify possible attacks.

Attack behaviors. Decoupling the control layer and the data layer and the design of centralized management of network behavior by the controller brings some specific new threats to SDN. For example, Santos et al. [34] used four machine learning methods, such as random forest, to detect SDN. As a result, there are three types of DDoS attacks: controller attack, flow table attack, and bandwidth attack. Among them, the classification performance of controller attacks is the worst, possibly because of important features that are similar to the detection of SDN attack types and normal traffic patterns. For example, the duration features of traffic in SDN controller attacks are similar to those of normal traffic. Although these features are widely used in traditional networks to detect different attacks, they cannot effectively detect SDN controllers—device attacks. In addition, common SDN attacks include saturation attacks from the data layer to the control layer, link flooding attacks, and flow-rule flooding attacks. Abbas et al. [50] pointed out that there are different correlations between the traffic features and various attack behaviors in the SDN environment and analyzed six types of attack behaviors for detecting DoS, DDoS, brute force cracking, exploration, botnets, and web applications. For example, traffic duration has the greatest impact on distinguishing normal traffic from brute-force, probe, botnet, and Web application attacks, whereas the total deviation in the reverse packet sending time has the greatest impact on identifying DDoS attacks. Therefore, the dataset used to train the intrusion detection system must be collected in an SDN environment, various possible attack behaviors for the network architecture must be assumed, and the relevant features of these attack behaviors must be recorded to learn and recognize the type of attack effectively.

4.3.2. Detection Differences of between Active Attacks and Passive Attacks

The proposed model is trained against active attacks, such as DoS attacks. Although it may not be directly applicable to passive attacks such as eavesdropping, there may still be signatures or traffic analysis that can indirectly help detect passive attacks. We illustrate the differences between active and passive attack detection from three perspectives: datasets, methods, and measures.

1. Datasets. Actively attacked datasets typically include situations in which an attacker actively modifies or interferes with the data. In contrast, passive attacks involve the unauthorized interception of data without actively modifying or destroying data. Therefore, the period for collecting samples, the attack behavior identification, and labeling rules of markers must be considered.
2. Methods. The common method of detecting active attacks is the use of machine-learning classifiers or rule-based methods for anomaly detection. In contrast, passive attacks typically use traffic analysis to detect snooping attacks or apply encryption technology to protect data. Even if an attacker intercepts data while it is encrypted, extracting meaningful information is difficult. For example, An and Yang [51] formulated a new framework for opacity for distributed state estimation to meet the confidentiality requirement. In addition, they proposed two opacity-enhancing algorithms against the intruder models with different eavesdropping capacities for a subset of nodes.
3. Measures. The evaluation of active attacks usually involves testing the detection rate, accuracy rate, and efficiency of the training model for known attack patterns. Public training and testing datasets are typically used to provide the recognition and processing capabilities of a model for specific attack types. In contrast, passive attacks involve monitoring the accuracy of traffic patterns, the security of encryption algorithms, and the ability to detect anomalous activity. Additionally, known passive attack patterns or synthetic test cases can be used to evaluate the ability of a defense system to detect and protect against snooping attacks.

4.3.3. Challenges of Methods

Although the proposed model did not yield breakthrough performance, it still illustrated novel concepts and provided improvements over existing methods. Combining transfer learning and meta-learning techniques is challenging because of the following three aspects:

1. Model complexity. Both transfer learning and meta-learning are mathematical model optimization techniques. The proposed architecture combines the existing Reptile model with the concept of transfer learning. The challenges involve parameter tuning to balance predictive performance with efficiency as well as preparing source and target domain datasets.
2. Heterogeneity. Transfer learning and meta-learning usually operate at different levels of abstraction, and their goals are not identical; transfer learning focuses on leveraging knowledge from the source domain to improve learning in the target domain; whereas meta-learning aims to learn efficiently through a sequence of tasks. The proposed Reptitle-TL model considers the correlation between the source and target domains based on transfer learning. Reptile model was then used to quickly obtain the best parameters for the transfer learning model.
3. Architecture optimization. For model architecture design, combining transfer learning and meta-learning models requires the effective integration of two components, and developing a unified architecture that allows seamless integration and avoids conflicts or redundancies is challenging. Regarding effectiveness optimization, transfer learning focuses on fine-tuning pre-trained models for specific tasks, whereas meta-learning aims to learn knowledge to quickly adapt to new tasks. As shown in Figure 6, the transfer learning model first pre-trained the source domain model, after which meta-learning quickly obtained the parameters required for new tasks

(Figure 5). By updating these parameters, transfer learning enables them to fine-tune the target domain model for optimization.

5. Conclusions

To effectively detect anomalous threats in an SDN environment, we used deep learning (CNN) models, two transfer learning models (Convnet-TL and Reptile-TL), and three meta-learning models (MAML, Meta-SGD, and Reptile) to evaluate the performance. The experiments simulated three types of practical issues: unknown attacks, small samples, and class imbalance. Our Reptile-TL model combined two-stage tasks: abnormal detection and multi-class attack classification. Our method retained the advantages of deep learning and transfer learning, first using the CNN trained in the source domain as the feature extractor of the target domain and then adopting the Reptile meta-learning training method to obtain suitable initial parameters for training. Thus, our model outperformed the other models in three cases: unknown attack, small number of samples, and class imbalance.

The experimental results also showed that general deep-learning methods achieved stable and good performance with sufficient training samples. By contrast, ConvNet-TL and Reptile-TL can be used for unknown attacks and intrusion detection with few training samples. Transfer learning and meta-learning methods offer several advantages; however, minority classification remains challenging. As the minority and majority classes are trained together and exist in a data distribution space, the classifiers are easily biased and misjudged for majority classes with larger training samples.

In future research, we will consider resampling to minimize the difference in the number of samples between classes to alleviate class imbalance issues. We will also explore the critical features of attack types to effectively improve the performance of unknown attack detection and strengthen the multi-class classification performance for small samples.

Author Contributions: H.-M.C. conducted the methodology, concept, and writing—original draft preparation. L.-J.Y. conducted the model training and programming. All authors have read and agreed to the published version of the manuscript.

Funding: This research is grateful for the support of the NSTC 111-2221-E-033-052.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The SDN public datasets can be downloaded from the websites of the original authors. For the experimental results of this manuscript, you can contact the corresponding author.

Acknowledgments: This research is grateful for the support of the National Defense Science and Technology Academic Collaborative Research Project in 2023.

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

References

1. Ahmed, L.A.H.; Hamad, Y.A.M. Machine learning techniques for network-based intrusion detection system: A survey paper. In Proceedings of 2021 National Computing Colleges Conference (NCCC), Taif, Saudi Arabia, 27–28 March 2021; pp. 1–7. [CrossRef]
2. Uğurlu, M.; Doğru, İ.A. A survey on deep learning based intrusion detection system. In Proceedings of the 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, 11–15 September 2019; pp. 223–228. [CrossRef]
3. Atay, I. Intrusion detection with probabilistic neural network: Comparative analysis. In Proceedings of the International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18), Safranbolu, Karabuk, Turkey, 11–13 May 2018.
4. Niu, S.; Liu, Y.; Wang, J.; Song, H. A decade survey of transfer learning (2010–2020). *IEEE Trans. Artif. Intell.* **2020**, *1*, 151–166. [CrossRef]
5. Hewlett Packard Enterprise (HPE) Aruba Networking, 5 Networking Predictions For 2022. Available online: https://www.arubanetworks.com/resource/5-networking-predictions-for-2022/ (accessed on 26 February 2023).
6. Chowdhury, N.M.M.K.; Boutaba, R. A survey of network virtualization. *Comput. Networks* **2010**, *54*, 862–876. [CrossRef]

7. Bhamare, D.; Jain, R.; Samaka, M.; Erbad, A. A survey on service function chaining. *J. Netw. Comput. Appl.* **2016**, *75*, 138–155. [CrossRef]
8. Ali, J.; Jhaveri, R.H.; Alswailim, M.; Roh, B.-H. ESCALB: An effective slave controller allocation-based load balancing scheme for multi-domain SDN-enabled-IoT networks. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 101566. [CrossRef]
9. Rahouti, M.; Xiong, K.; Xin, Y.; Jagatheesaperumal, S.K.; Ayyash, M.; Shaheed, M. SDN security review: Threat taxonomy, implications, and open challenges. *IEEE Access* **2022**, *10*, 45820–45854. [CrossRef]
10. Jiménez, M.B.; Fernández, D.; Rivadeneira, J.E.; Bellido, L.; Cárdenas, A. A survey of the main security issues and solutions for the SDN architecture. *IEEE Access* **2021**, *9*, 122016–122038. [CrossRef]
11. Mubarakali, A.; Alqahtani, A.S. A survey: Security threats and countermeasures in software-defined networking. In Proceedings of the 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), Kahului, HI, USA, 14–17 March 2019; pp. 180–185. [CrossRef]
12. Elsayed, M.S.; Le-Khac, N.-A.; Jurcut, A.D. InSDN: A novel SDN intrusion dataset. *IEEE Access* **2020**, *8*, 165263–165284. [CrossRef]
13. Liyanage, M.; Braeken, A.; Jurcut, A.D.; Ylianttila, M.; Gurtov, A. Secure communication channel architecture for software defined mobile networks. *Comput. Netw.* **2017**, *114*, 32–50. [CrossRef]
14. Kreutz D.; Fernando, M.V.R.; Verissimo, P. Towards secure and dependable software-defined networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13), New York, NY, USA, 16 August 2013; pp. 55–60.
15. Kumar, R.; Lal, S.P.; Sharma, A. Detecting denial of service attacks in the cloud. In Proceedings of the 2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Auckland, New Zealand, 13 October 2016; pp. 309–316. [CrossRef]
16. Liao, H.-J.; Lin, C.-H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24. [CrossRef]
17. Kumar, M.; Hanumanthappa, M.; Kumar, T.V.S. Intrusion detection system using decision tree algorithm. In Proceedings of the 2012 IEEE 14th International Conference on Communication Technology, Chengdu, China, 9–11 November 2012; pp. 629–634.
18. Rathore, M.M.; Saeed, F.; Rehman, A.; Paul, A.; Daniel, A. Intrusion detection using decision tree model in high-speed environment. In Proceedings of the 2018 International Conference on Soft-Computing and Network Security (ICSNS), Coimbatore, India, 14–16 February 2018; pp. 1–4.
19. Ali, J.; Roh, B.-H.; Lee, B.; Oh, J.; Adil, M. A machine learning framework for prevention of software-defined networking controller from DDoS attacks and dimensionality reduction of big data. In Proceedings of 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 21–23 October 2020; pp. 515–519. [CrossRef]
20. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT. *Sensors* **2017**, *17*, 1967. [CrossRef]
21. Roy, S.S.; Mallik, A.; Gulati, R.; Obaidat, M.S.; Krishna, P.V. A deep learning based artificial neural network approach for intrusion detection. In Proceedings of the International Conference on Mathematics and Computing (ICMC), Haldia, India, 17–21 January 2017.
22. Kim, J.; Shin, N.; Jo, S.Y.; Kim, S.H. Method of intrusion detection using deep neural network. In Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, Republic of Korea, 13–16 February 2017; pp. 313–316.
23. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep learning approach for intelligent intrusion detection system, *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
24. Assis, M.V.; Carvalho, L.F.; Lloret, J.; Proença, M.L. A GRU deep learning system against attacks in software defined networks. *J. Netw. Comput. Appl.* **2021**, *177*, 102942. [CrossRef]
25. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [CrossRef]
26. Wu, P.; Guo, H.; Buckland, R. A transfer learning approach for network intrusion detection. In Proceedings of the 2019 4th IEEE International Conference on Big Data Analytics (ICBDA), Suzhou, China, 15–18 March 2019; pp. 281–285.
27. Rodríguez, E.; Valls, P.; Otero, B.; Costa, J.J.; Verdú, J.; Pajuelo, M.A.; Canal, R. Transfer-learning-based intrusion detection framework in IoT networks. *Sensors* **2022**, *22*, 5621. [CrossRef] [PubMed]
28. Moustafa, N. The Bot-IoT Dataset. *IEEE Dataport*. 16 November 2019. Available online: https://ieee-dataport.org/documents/bot-iot-dataset (accessed on 26 February 2023).
29. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
30. Dai, W.; Yang, Q.; Xue, G.; Yu Y. Boosting for transfer learning, In Proceedings of the 24th International Conference Machine Learning, Corvallis, OR, USA, 20–24 June 2007; pp. 193–200. [CrossRef]
31. Wang, T.; Lv, Q.; Hu, B.; Sun, D. A few-shot class-incremental learning approach for intrusion detection. In Proceedings of the 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; pp. 1–8.
32. Singla, A.; Bertino, E.; Verma, D. Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 12–15 June 2019; pp. 69–74. [CrossRef]

33. Dhillon, H.; Haque, A. Towards network traffic monitoring using deep transfer learning. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 12 December–1 January 2021; pp. 1089–1096. [CrossRef]
34. Santos, R.R.D.; Viegas, E.K.; Santin, A.O. A reminiscent intrusion detection model based on deep autoencoders and transfer learning. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
35. Sameera, N.; Shashi, M. Deep transductive transfer learning framework for zero-day attack detection. *ICT Express* **2020**, *6*, 361–367. [CrossRef]
36. Dhanabal, L.; Shantharajah, S.P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.
37. Duy, D.T.; Khoa, N.H.; Hiep, H.; Tuan, N.B.; Hoang, H.D.; Hien, D.T.T.; Pham, V.-H. *A Deep Transfer Learning Approach for Flow-Based Intrusion Detection in SDN-Enabled Network*; IOS PRESS: Amsterdam, The Netherlands, 2021; Volume 337, pp. 327–339. [CrossRef]
38. 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]
39. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
40. Das, A. A deep transfer learning approach to enhance network intrusion detection capabilities for cyber security. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 843–855. [CrossRef]
41. Taghiyarrenani, Z.; Fanian, A.; Mahdavi, E.; Mirzaei, A.; Farsi, H. Transfer learning based intrusion detection. In Proceedings of the 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 25–26 October 2018; pp. 92–97.
42. Phan, T.V.; Sultana, S.; Nguyen, T.G. Q-TRANSFER: A novel framework for efficient deep transfer learning in networking. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 146–151.
43. Gamal, M.; Abbas, H.M.; Moustafa, N.; Sitnikova, E.; Sadek, R.A. Few-shot learning for discovering anomalous behaviors in edge networks. *Comput. Mater. Contin.* **2021**, *69*, 1823–1837. [CrossRef]
44. Khoa, N.H.; Hiep, H.; Tuan, N.B.; Hoang, H.D.; Hien, D.T.T.; Pham, V.H. A deep transfer learning approach for flow-based intrusion detection in SDN-enabled network. In Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SOMET 2021), online, 21–23 September 2021; pp. 327–339.
45. Wang, D.; Chen, X.; Chen, D. DDoS detection method based on instance transfer learning. In Proceedings of the 2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP), Nanjing, China, 9–11 July 2021; pp. 1053–1058. [CrossRef]
46. Xu, C.; Shen, J.; Du, X. A method of few-shot network intrusion detection based on meta-learning framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552. [CrossRef]
47. Yu, Y.; Bian, N. An intrusion detection method using few-shot learning. *IEEE Access* **2020**, *8*, 49730–49740. [CrossRef]
48. Banton, M.; Shone, N.; Hurst, W.; Shi, Q. Intrusion detection using extremely limited data based on SDN. In Proceedings of the 2020 IEEE 10th International Conference on Intelligent Systems (IS), Varna, Bulgaria, 28–30 August 2020; pp. 304–309.
49. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 22–24 January 2018; pp. 108–116. [CrossRef]
50. Abbas, N.; Nasser, Y.; Shehab, M.; Sharafeddine, S. Attack-specific feature selection for anomaly detection in software-defined networks. In Proceedings of the 2021 3rd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), Agadir, Morocco, 3–5 December 2021; pp. 142–146. [CrossRef]
51. An, L.; Yang, G.-H. Enhancement of opacity for distributed state estimation in cyber-physical systems. *Automatica* **2022**, *136*, 110087. [CrossRef]