electronics

MDPI

Article

Deep Learning for Network Intrusion Detection in Virtual Networks

Daniel Spiekermann [1,2,†], Tobias Eggendorfer [2,3,†] and Jörg Keller [2,*,†]

[1] Faculty of Computer Science, Dortmund University of Applied Sciences and Arts, 44227 Dortmund, Germany; daniel.spiekermann@fh-dortmund.de
[2] Faculty of Mathematics and Computer Science, FernUniversität in Hagen, 58097 Hagen, Germany
[3] Faculty of Computer Science, Technische Hochschule Ingolstadt, 85049 Ingolstadt, Germany; tobias.eggendorfer@fernuni-hagen.de
* Correspondence: joerg.keller@fernuni-hagen.de
† These authors contributed equally to this work.

Abstract: As organizations increasingly adopt virtualized environments for enhanced flexibility and scalability, securing virtual networks has become a critical part of current infrastructures. This research paper addresses the challenges related to intrusion detection in virtual networks, with a focus on various deep learning techniques. Since physical networks do not use encapsulation, but virtual networks do, packet analysis based on rules or machine learning outcomes for physical networks cannot be transferred directly to virtual environments. Encapsulation methods in current virtual networks include VXLAN (Virtual Extensible LAN), an EVPN (Ethernet Virtual Private Network), and NVGRE (Network Virtualization using Generic Routing Encapsulation). This paper analyzes the performance and effectiveness of network intrusion detection in virtual networks. It delves into challenges inherent in virtual network intrusion detection with deep learning, including issues such as traffic encapsulation, VM migration, and changing network internals inside the infrastructure. Experiments on detection performance demonstrate the differences between intrusion detection in virtual and physical networks.

Keywords: virtual network; intrusion detection; intrusion prevention; deep learning

check for updates

Citation: Spiekermann, D.; Eggendorfer, T.; Keller, J. Deep Learning for Network Intrusion Detection in Virtual Networks. *Electronics* **2024**, *13*, 3617. https://doi.org/10.3390/electronics13183617

Academic Editor: Rajendra V. Boppana

Received: 24 July 2024
Revised: 3 September 2024
Accepted: 6 September 2024
Published: 11 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

More and more applications have moved to the cloud or at least into virtualized environments and hence need, like any computer network, protection from attacks through established security measures like firewalls and network intrusion detection. Yet, classic IT security measures such as firewalling, network intrusion detection and prevention, as well as post-incident forensics rely on technologies that were up to date when data centers were on site. Virtual machines in virtualized environments can migrate frequently, and virtual routers can be even added or removed when the virtual network is reconfigured, which is a frequent operation under the control of the tenant (not the provider) to adapt its structure to actual load and tasks. This means that virtual networks could change network flows in no time, leaving no fixed place where the intrusion detection system can connect, and thus pushing it outside the network where it can only observe encapsulated packets from the virtual network (cf. Section 2 for details). This makes intrusion detection and network forensics a significant topic in both virtual and physical networks. In both types of networks, intrusion detection allows the identification of attacks on systems attached to the network and potentially prevents them. Examples of attacks detected include, but are not limited to, SQL injections and buffer overflows, but also preparation of attacks such as port scanning. While in physical networks, intrusion detection is considered state of the art to protect systems ranging from web shops to database servers or home automation, in virtual networks, intrusion detection systems are not yet available. Therefore, a necessity for further research is given.

Recent work has analyzed how network forensics, especially live forensics, could be conducted in virtual networks [1]; however, this is still a very complex and challenging environmentwhich often goes unnoticed, such as in [2]. Some authors have considered network intrusion detection in software-defined networks [3], which potentially could provide a base for investigation into intrusion detection in virtual networks. Others discuss whether deep learning could provide benefits in network intrusion detection [4,5].

Despite the importance of this topic, research on intrusion detection in virtual networks remains limited. Based on our previous analysis on virtual network forensics, especially the monitoring of network traffic [1], we investigate network intrusion detection via deep learning in virtual networks. We implement a network data processing pipeline and compare deep learning-based classification of network flows in classic and virtualized networks. We can demonstrate that the latter have worse classification accuracy. Re-learning with virtualized network data can improve accuracy, but still not to the level that is present for non-virtualized networks. Hence, specific consideration of virtualization in intrusion detection is a must, and further research seems necessary.

The remainder of this article is structured as follows. In Section 2, we summarize background information on virtual networks, network intrusion detection, and machine learning. Section 3 discusses related work. Section 4 presents our experimental methodology, while Section 5 details our evaluation and Section 6 summarizes our results. Finally, Section 7 gives conclusions and an outlook to future work.

2. Background

2.1. Virtual Networks

In the rapidly evolving landscape of network infrastructure, virtual networks have become indispensable for achieving flexibility, scalability, and efficient resource utilization. These networks abstract and virtualize underlying physical resources, enabling multiple virtual networks to coexist on the same physical infrastructure. However, the complexity introduced by virtualization poses significant challenges, particularly in terms of network performance and security. Common virtual networks are separated in the two following areas:

Underlay Network: A robust and scalable IP/MPLS (Internet Protocol/Multi-Protocol Label Switching) underlay network is essential to support every kind of virtual network on top of this network. Proper network design, including adequate routing and QoS (Quality of Service) mechanisms, is crucial to ensure optimal performance. The underlay network provides the basic infrastructure upon which virtual networks are built, and its reliability directly impacts the effectiveness and efficiency of the overlay networks.

Overlay Design: The virtual environment runs completely on top of the underlay network in the so-called overlay network, making use of encapsulation protocols. The design, including addressing schemes, route distribution policies, and redundancy mechanisms, is critical for a reliant infrastructure. Beyond the network itself being virtual, the machines connected by the network can also be virtual machines (VMs), which can be quickly started, shut down, or migrated to other parts of the network, leading to frequent changes in network configuration, like routes. Even the network hardware such as routers can be virtual, i.e., provided as software or a virtual machine. Modern techniques like an EVPN (Ethernet Virtual Private Network, see below) provide more flexibility than prior implementations like VXLAN-networks (Virtual Extensible Local Area Network) without a central control plane. These advancements in overlay design enhance the manageability and robustness of virtual networks, ensuring they can meet the dynamic demands of contemporary network environments.

EVPN: An Ethernet Virtual Private Network (EVPN) is a cutting-edge technology that has gained significant traction in modern network infrastructures, particularly in data center environments. An EVPN provides a scalable and efficient solution for interconnecting geographically distributed data centers while offering advanced capabilities for multi-tenancy, mobility, and network virtualization.

An EVPN is designed to address the limitations of traditional Layer 2 VPNs (Virtual Private Networks) such as Virtual Private LAN Service (VPLS). By leveraging BGP (Border Gateway Protocol) as the control plane protocol, an EVPN offers a more scalable and flexible approach to building Layer 2 and Layer 3 virtual private networks over an IP/MPLS infrastructure [6].

2.2. Network Intrusion Detection

Network intrusion detection systems (NIDSs) are used to detect attacks, including network intrusions, extrusions, and other anomalies. Most NIDS analyze live network traffic in order to identify potential issues and alert network administrators or, if used as an intrusion prevention system (IPS), adjust firewall rules to block unwanted traffic.

Detection could be based on simple rules that could be triggered without analyzing the packet content, e.g., a sequence of ports being accessed from a single IP as an indicator for a port scan. Rules based on a content analysis often struggle with encryption, i.e., to detect an SQL-injection, the NIDS would either need to have a sensor on the target system, analyzing packets once de-crypted, or be able to break encryption. The latter was suggested by [7], which, however, poses a security risk [8,9]. Ref. [10] therefore tried to analyze encrypted packets for suspicious patterns.

An alternative to applying rules to the header or content of a packet is to detect anomalies. These could match predefined rules, which need to match the use case of the system, or could, e.g., be generated through machine learning, which, however, needs a network not under attack but under regular operation for the system to identify the "normal". Anomaly detection is a heuristic process, with a rather high risk for false positives and false negatives [11,12]. With virtual networks, this becomes more complex due to the needed encapsulation of the packets [13].

2.3. Machine Learning and Deep Learning

With *machine learning* (ML), the computer engineers a decision-making rule set, rather than the software developer. Several models for ML exist: With supervised learning, the dataset to learn with is pre-labeled. With un-supervised learning, it is not. Self-supervised learning then provides input-label pairs [14]. An implementation of self-supervised learning is *deep learning* (DL).

DL employs a multi-layer neural network with a long credit assignment path (CAP). The CAP measures how long the chain of transformation in the neural network is. However, there is no precise value for the CAP to differentiate DL from "non-deep" learning [15]. In this research, we used a Convolutional Neural Network (CNN) architecture for network intrusion detection in virtualized environments. CNNs are commonly used for tasks involving spatial data and have shown effectiveness in capturing patterns within sequences of data, making them a suitable choice for analyzing network traffic [16,17]. The choice of a CNN for this study was driven by several factors like pattern recognition in network traffic based on the representation of packets as flows, as well as scalability, performance, and previous usage in similar research like [13]. By selecting a CNN, our research aims to leverage the strengths of deep learning in capturing complex patterns within network traffic and applying these insights to the challenge of intrusion detection in virtualized environments. Future work may explore the integration of other deep learning models, such as Recurrent or Deep Neural Networks (RNNs/DNNs), to further enhance detection capabilities, particularly for time-dependent patterns in network traffic.

3. Related Work

Network forensics as well as intrusion detection could be performed on either live data or on generated logs. Rajesh et al. propose to analyze virtual networks based on generated log files [18], using machine learning and elastic search. However, this approach is only usable in post-mortem analysis. As it requires access to log files, it cannot analyze live traffic, and thus merely complements our work, which can be used to analyze live traffic or

recorded network traffic, and solely uses the network data themselves, without needing additional access to log files.

Live traffic analysis is performed by Wang et al. in [19] by modifying a virtual bridge so that it provides an option to either forward traffic for analysis or analyze it itself. Although the concept is feasible, the implementation described in the paper is highly dependant on the system used for virtualization, and thus it is hard to adapt it to other environments. Therefore, it lacks the general perspective that our research considers.

Modi and Patel adapt the popular network intrusion detection system Snort to analyze traffic in the cloud by applying associative classifiers to its detection rules [20]. To do so, on each VM a NIDS sensor is installed. This is a feasible setup for someone protecting their own assets in a cloud. However, this approach is not feasible for identifying malicious traffic in virtual networks without access to each VM, which is the focus of our research.

In contrast, Chung et al. add a monitoring agent to each virtual machine host [21], assuming that they should be able to unpack network data and analyze them. The analysis, however, does not focus on network data, but is more on the behavioral side of the virtual machine, including network anomalies. While again this is an efficient idea to monitor a VM's state and detect potential intrusions, it does not fully comply with the direction of our research, which is on network intrusion detection and not on the deduction of network intrusion from machine behavior.

Dina and Manivannan survey ML-based intrusion detection approaches [22]. They state intrusion detection in virtualized environments as an open research question (see Section 5.1 in [22]), partly because of the lack of appropriate datasets: the datasets that they list in Section 2.3 of [22] are rather old, and therefore do not comprise virtualized network traffic, which is the focus of our research.

Zhang et al. consider the generation of attack graphs from virtual networks to correlate them with intrusion detection [23]. However, they do not consider the dynamic nature of virtual networks with frequent migrations and reconfigurations, and thus are quite far from our work.

Zamrii et al. investigate the maintenance of functionality of an information system when it is realized as a virtual network and can be reconfigured when an intrusion detection is notified [24]. However, they do not investigate the detection of a network intrusion itself, which is the focus of our work, but assume this as given.

Srivinas et al. [25] investigate intrusion detection in a cloud; yet, they mainly consider a compute cloud where users place tasks to be executed, and their detection is based on VM migration characteristics. They evaluate their proposal with the UNSW's BoT-IoT dataset 2020, which means that their focus is different from ours, too. In contrast to our research, they do not consider network data for intrusion detection, so the encapsulation of data packets in virtual networks does not play a role in their approach.

The restricted number of related works, especially in recent years, seems to indicate that the topic of network intrusion detection in virtual networks has been largely overlooked so far despite the spread of virtual networks in practice, which underlines the importance of the current research.

4. Methodology

To evaluate the behavior of a deep learning network when faced with changed network data, we first need to train the neural network. As training data, we use recorded network data as done in previous studies [26,27]. However, in contrast to other studies, we captured this data ourselves, with and without attacks, and only complemented them by network data from published datasets. To do so, we created a pipeline, as shown in Figure 1. The different steps process the data in order to create a deep learning network with a good accuracy when detecting attacks or intrusions in the network. We focus on the analysis of well-known deep learning models without a specific preparation for virtual networks. By doing this, we are able to analyze the detection rate based on models trained with common

network data without any virtual network protocols. Because of this, we implement a pipeline similar to [26,27].

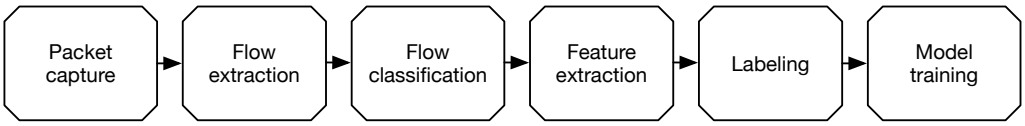


Figure 1. Data processing pipeline.

4.1. Data Acquisition and Preparation

The methodology employed for analyzing network traffic and the detection of possible intrusions utilizes a robust deep learning framework facilitated by TensorFlow. Our approach begins with the collection of network traffic data. Once collected, the data are pre-processed and redundant or irrelevant information is filtered out, and incompatible data are transformed into a common format suitable for our purposes. This standardization ensures the consistency and reliability of the dataset for the following tasks, like flow and feature extraction. To implement the first model for network-based intrusion detection, we used network traffic captured at various positions, like home networks, separated university subnets, internet servers, and cloud environments with internal networks and distributed locations. The overall amount of network traffic encompass 394 million network packets with various network protocols like TCP, UDP, and QUIC as transport protocols, IPv4 and IPv6 as network protocols, application protocols like DNS, DHCP, HTTP(s), FTP, Tor, MySQL, IMAP, and SMTP, and local network protocols like MDNS, Bonjour, ARP, and NDP. We additionally performed various network-based attacks with Metasploit and added various pcap-files with malicious traffic published in datasets like UNSW-NB15 [28]. We converted non-virtualized traffic into traffic on virtual networks with the tool Encapcap [29] (see also Section 5.1), which obtains a script that specifies, e.g., which physical network will be changed into a virtual network and when and how virtual machines might migrate.

4.2. Flow Extraction and Classification

Our approach focuses on the detection of intrusion based on network flows. Therefore, we convert the packets in the pcap-file into a flow comprising relevant information related to the flow, like duration, time deltas between the single packets, and used flags if the packets are transferred via TCP. We used the python library nfstream (details can be found at https://www.nfstream.org/, (accessed on 5 September 2024)) to read the different pcap-files and extract the flows of each file. These flows were analyzed and, if relevant, processed to the feature extraction job. Irrelevant flows like internal-only traffic were removed from the list of flows during the flow classification phase. The process of classification is different from the subsequent step of labeling. Classification removes irrelevant data from the list of flows, whereas labeling defines the relevance (benign/malicious) of the remaining flows.

4.3. Feature Extraction

After the pre-processing, the next step involves extracting relevant features critical for identifying potential intrusions. These features include, but are not limited to, IP addresses, port numbers, protocols, and time-dependant values. The definition of relevant features is part of current research, like [30,31], but to our best knowledge, there is no existing feature set for virtual network packets. Because of this, we chose a list of features, which works well in combination with nfstream [32,33], as shown in Table 1. The processed data are then used to construct a well-defined dataset, which forms the foundation for the training of the deep learning process.

Table 1. List of features and data types.

Feature	Data Type
src_ip	Categorical (String)
src_port	Numeric (Integer)
dst_ip	Categorical (String)
dst_port	Numeric (Integer)
protocol	Categorical (String)
bidirectional_packets	Numeric (Integer)
bidirectional_bytes	Numeric (Integer)
bidirectional_duration_ms	Numeric (Float)
src2dst_duration_ms	Numeric (Float)
src2dst_packets	Numeric (Integer)
src2dst_bytes	Numeric (Integer)
dst2src_duration_ms	Numeric (Float)
dst2src_packets	Numeric (Integer)
dst2src_bytes	Numeric (Integer)
bidirectional_min_ps	Numeric (Float)
bidirectional_mean_ps	Numeric (Float)
bidirectional_stddev_ps	Numeric (Float)
bidirectional_max_ps	Numeric (Float)
src2dst_min_ps	Numeric (Float)
src2dst_mean_ps	Numeric (Float)
src2dst_stddev_ps	Numeric (Float)
src2dst_max_ps	Numeric (Float)
dst2src_min_ps	Numeric (Float)
dst2src_mean_ps	Numeric (Float)
dst2src_stddev_ps	Numeric (Float)
dst2src_max_ps	Numeric (Float)
bidirectional_min_piat_ms	Numeric (Float)
bidirectional_mean_piat_ms	Numeric (Float)
bidirectional_stddev_piat_ms	Numeric (Float)
bidirectional_max_piat_ms	Numeric (Float)
src2dst_min_piat_ms	Numeric (Float)
src2dst_mean_piat_ms	Numeric (Float)
src2dst_stddev_piat_ms	Numeric (Float)
src2dst_max_piat_ms	Numeric (Float)
dst2src_mean_piat_ms	Numeric (Float)
dst2src_stddev_piat_ms	Numeric (Float)
dst2src_max_piat_ms	Numeric (Float)
bidirectional_ack_packets	Numeric (Integer)
bidirectional_psh_packets	Numeric (Integer)
bidirectional_rst_packets	Numeric (Integer)
bidirectional_fin_packets	Numeric (Integer)

4.4. Labeling

The extracted features of each flow are stored in a csv-file, which simplifies the subsequent training with TensorFlow. The labeling process for the dataset involved tagging each flow as either "benign" or "malicious" based on its source. Traffic generated by normal operations was labeled as benign, while traffic associated with the simulated attacks was labeled as malicious. By doing this, normal traffic flows, such as legitimate HTTP requests, DNS queries, and internal VM communications, were labeled as benign, whereas flows associated with reconnaissance, exploitation, and DoS attacks were labeled as malicious. This labeled dataset forms the basis for training the deep learning model, ensuring that it can distinguish between normal and anomalous traffic in a virtualized environment.

4.5. Model Training

With the dataset prepared, a deep learning model is constructed using TensorFlow. This model leverages a CNN based on different layers of Dense, BatchNormalization, and Dropout chosen for its proficiency in handling sequence data, which is typical of network traffic. The architecture consists of several layers optimized for the detection of anomalies in network traffic. The specific configuration of layers and hyperparameters used during training is detailed below to ensure reproducibility.

- Input Layer: The input layer receives the feature set extracted from the network flows. The number of input nodes corresponds to the dimensionality of the feature set, which is 38 features as defined in the feature extraction step.
- Hidden Layers:
  - Layer 1: A fully connected dense layer with 32 neurons and ReLU activation function.
  - Layer 2: A fully connected dense layer with 16 neurons and ReLU activation function.
  - Layer 3: Batch normalization to normalize the outputs and stabilize the learning process.
  - Layer 4: Another dense layer with 16 neurons and ReLU activation function.
- Output Layer: A single neuron with a sigmoid activation function is used for binary classification, outputting the probability that a given flow is malicious.

The model is trained on the curated dataset, where it learns to discern between normal operations and potential intrusions. In order to analyze the behavior of known models and algorithms when faced with virtual protocols, we did not add further hyperparameters or layers to our deep learning model, but implemented it similar to [13]. As a result, the deployed hyperparameters are as follows:

- Optimizer: Adam optimizer was selected due to its efficiency and effectiveness in handling sparse gradients, with a learning rate of 0.001.
- Loss Function: Binary cross-entropy was employed to measure the loss between predicted and actual labels, suitable for binary classification tasks.
- Batch Size: A batch size of 64 was chosen based on experimental validation, balancing memory constraints and model convergence speed.
- Epochs: The model was trained for 50 epochs, which provided sufficient time for convergence without overfitting, as validated by the training and validation loss curves.
- Regularization: Dropout layers were not included as overfitting was controlled through careful monitoring of validation performance. Future work may explore dropout or L2 regularization if overfitting becomes evident in more complex datasets.

Figure 2 illustrates the increasing accuracy of our training model with increasing epochs.

Figure 3 illustrates the training loss as a function of the number of training iterations undertaken by the model. The data presented above reveal a significant decrease in training loss together with an increase in the number of epochs. This trend suggests that the model was progressively learning and enhancing its performance over time.

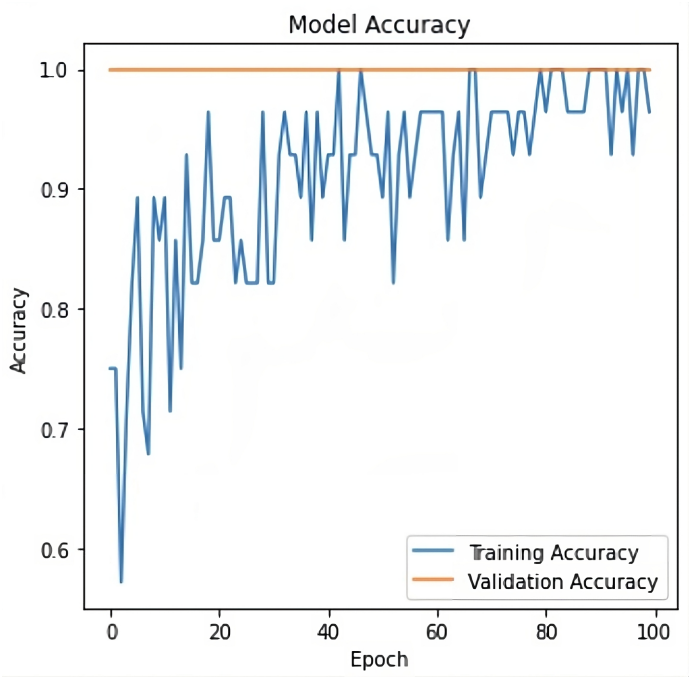


Figure 2. Increasing accuracy of our training model.

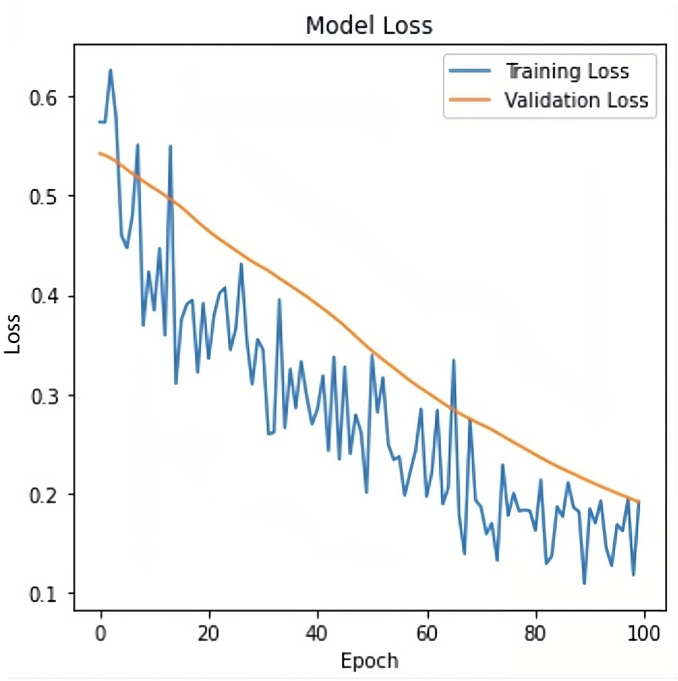


Figure 3. Loss of training model with increasing epochs.

4.6. Accuracy

Post-training, the model's effectiveness is evaluated through testing against a separate set of data not previously exposed to the model. This phase assesses the model's accuracy, providing insights into its performance and its capability to generalize across different network environments. Adjustments and optimizations are made based on the outcomes of this evaluation to enhance the model's predictive accuracy and reduce false positives.

Our model reaches an average accuracy of 97.95% depending on different input files, which provides a sufficient detection rate when classifying network flows as benign or malicious.

Such a detection rate is similar to the models discussed in [34,35]. The further metrics of our model are similar to these models, too. Our model reaches average values for precision of 98.14% and recall of 99.84%. Table 2 shows the confusion matrix of one input file with a flow count of 1,342,112.

Table 2. Confusion matrix.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	721,840	1140
	Negative	13,670	605,462

5. Evaluation

After the initial process of creating a deep learning model, we evaluate the model with network data from virtual networks. In normal use, the input of the deep learning model could come from live traffic or from recorded data. To implement our approach for live traffic analysis, NFStream can capture and extract our defined features from real-time traffic. The features are then fed into our model for an in-time classification. The system can be deployed across virtual machines or on uplinks to different network segments. This approach is similar to [36,37] and provides flexibility and robustness for a continuous traffic analysis.

For the purpose of our evaluation, i.e., to achieve comparable results, we convert the existing network traffic data (cf. Section 4) to virtualized network traffic to compare the resulting data with the trained model. Figure 4 shows the pipeline used for the evaluation of transformed network packets.

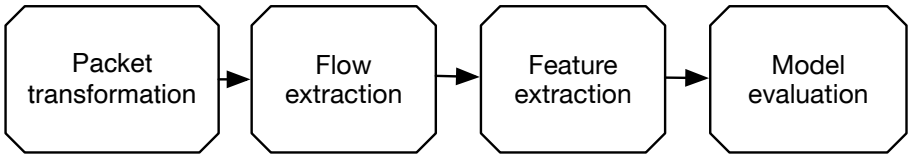


Figure 4. Transformed data processing pipeline.

5.1. Transformation

We use Encapcap [29] to transform the single network packets from *normal* to virtualized network traffic. Encapcap adds the relevant header data for encapsulation and takes care of fragmentation and other potential issues. This process is crucial to simulate the behavior of virtualized networks and analyze the performance of the intrusion detection system under these conditions. This results in the following data as shown in Figure 5 for normal traffic and the same, but now transformed, packet via VXLAN in Figure 6.

```
> Frame 2: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits)
> Ethernet II, Src: 0a:b4:df:27:c2:b0, Dst: 00:20:18:eb:ca:28
> Internet Protocol Version 4, Src: 192.168.40.1, Dst: 192.168.40.10
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (Offer)
```

Figure 5. Original network packet.

```
> Frame 2: 392 bytes on wire (3136 bits), 392 bytes captured (3136 bits)
> Ethernet II, Src: 11:22:33:44:55:66, Dst: aa:99:bb:88:cc:77
> Internet Protocol Version 4, Src: 192.168.44.1, Dst: 192.168.44.2
> User Datagram Protocol, Src Port: 4789, Dst Port: 4789
> Virtual eXtensible Local Area Network
> Ethernet II, Src: 0a:b4:df:27:c2:b0, Dst: 00:20:18:eb:ca:28
> Internet Protocol Version 4, Src: 192.168.40.1, Dst: 192.168.40.10
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (Offer)
```

Figure 6. VXLAN-transformed network packet.

The transformation of the network packet encompasses the addition of randomized data of simulated VTEPs. In our example, we use default values on layer 2 (i.e., MAC address), layer 3 (IPv4 addresses), and layer 4 (port number). The original network packet as shown in Figure 5 is then injected as the payload of the new VXLAN packet.

VXLAN is widely utilized in modern EVPN implementations, providing a scalable and efficient solution for interconnecting geographically distributed data centers. VXLAN allows for the extension of logical Layer 2 networks over a Layer 3 underlay network, thereby enhancing flexibility and scalability. This is particularly crucial in dynamic environments where virtual machine (VM) migrations and user customizations of internal networks frequently occur. The transformation of packets through VXLAN encapsulation effectively addresses challenges related to VM mobility and the adaptive configuration of internal network structures, ensuring seamless network connectivity and consistent performance. In this way, internal data might change together with the outer protocol information.

In addition to this, we further convert the network packets with a GENEVE (Generic Network Virtualization Encapsulation) encapsulation based on RFC 8926 [38], which results in packets structured as shown in Figure 7. The transformation of packets either to VXLAN as shown in Figure 6 or to GENEVE as shown in Figure 7 adds separate layers to the frames, each with different protocols and ports. The internal data remain unchanged; only the outer protocol data change.

```
> Frame 2: 392 bytes on wire (3136 bits), 392 bytes captured (3136 bits)
> Ethernet II, Src: 11:22:33:44:55:66, Dst: aa:99:bb:88:cc:77
> Internet Protocol Version 4, Src: 192.168.44.1, Dst: 192.168.44.2
> User Datagram Protocol, Src Port: 6081, Dst Port: 6343
> Generic Network Virtualization Encapsulation, VNI: 0x00005b
> Ethernet II, Src: 0a:b4:df:27:c2:b0, Dst: 00:20:18:eb:ca:28
> Internet Protocol Version 4, Src: 192.168.40.1, Dst: 192.168.40.10
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (Offer)
```

Figure 7. GENEVE-transformed network packet.

These transformed network packets are stored in a pcap-file, which is then used to extract the flows as discussed in Section 4.2. The created model is then used to classify the flows of the transformed network packets.

The goal of our simulation approach was to create a dataset that accurately represents the diverse and dynamic conditions found in real-world virtual networks. By capturing traffic from multiple environments, including cloud-based systems, and by incorporating scenarios unique to virtualized networks (e.g., VM migration), we aimed to ensure that our dataset reflects the complexities of modern network infrastructures. While the initial traffic capture was conducted in non-virtualized environments, the subsequent transformations and scenario injections were designed to bridge this gap. The use of Encapcap refined this process by converting the traffic into a virtual network context, applying encapsulation protocols to simulate the actual conditions under which an IDS would operate in a virtualized environment.

5.2. Classification

We use our created model without any adoptions or improvements to evaluate our model related to the transformed data.

We recognize a significant decrease in the detection rate when analyzing transformed network flows. Table 3 shows the results of the accuracy of the files with the different encapsulations.

Table 3. Metrics of our model with encapsulation protocols.

Encapsulation	Average Accuracy	Average Precision	Average Recall	F1-Score
None	97.95%	98.14%	99.84%	0.959
VXLAN	42.37%	94.97%	30.60%	0.439
GENEVE	35.66%	96.68%	17.08%	0.354
NVGRE	33.44%	96.89%	13.67%	0.384

5.3. Comparison with Well-Known Datasets

To evaluate our results, we used a well-known packet-based dataset with existing known attack. The number of datasets containing labeled network flows is high [39,40], but these datasets are mostly not eligible for a packet-based transformation process with Encapcap. Datasets with a full packet capture are CICDDoS2019 [41], which focuses on denial of service attacks, and Network TON_IoT [42] focusing on IoT-traffic, which is out of the scope of our research.

We use CSE-CIC-IDS 2018 [43], which provides a packet-based dataset containing various types of intrusion attacks like SQL-Injection, FTP attacks, bruteforce, or infiltration. The traffic of these attacks provides a better basis than datasets without these types of attacks or with a higher rate of benign traffic like UNSW-NB15.

Our model has an accuracy of 95.32% when analyzing the original flows extracted from the dataset, but after transforming the packets to encapsulated network traffic, the accuracy rate decreases depending on the protocol, as shown in Table 4.

Table 4. Accuracy of our model of the dataset CSE-CIC-IDS 2018.

Encapsulation	Average Accuracy	Average Precision	Average Recall
None	95.32%	92.45%	99.56%
VXLAN	41.85%	52.96%	37.55%
GENEVE	38.12%	44.80%	29.29%
NVGRE	39.29%	48.68%	31.71%

6. Summary of Results

In virtual network protocols such as VXLAN, GENEVE, and NVGRE, detection rates are lower than in non-virtualized environments. This can be attributed to the following inherent challenges and complexity:

- Encapsulated traffic;
- Higher levels of traffic variability;
- Optimization for higher network performance.

6.1. Encapsulated Traffic

Virtual network protocols encapsulate Layer 2 traffic over a Layer 3 network by adding additional headers and encapsulation layers. These additions might obscure the underlying traffic patterns. Detection algorithms fail at least partly to accurately parse and analyze the encapsulated traffic. This leads to a higher likelihood of false negatives and false positives. The differences between different virtual network protocols can be explained via their different structures.

VXLAN encapsulates Layer 2 traffic over a Layer 3 network using the 24-bit VNI. This process adds a fixed overhead to the packet structure but maintains a relatively simple

encapsulation scheme. In contrast, GENEVE is designed to be more flexible and extendable, supporting additional metadata in the form of optional TLVs (Type–Length–Value objects). This flexibility, while beneficial for certain networking use cases, introduces additional variability in packet structures, making it more challenging for intrusion detection systems to identify consistent patterns.

6.2. Higher Levels of Traffic Variability

Virtualized networks were introduced to support variability in networking, such as dynamic changes due to the frequent creation and deletion of virtual machines. They also support a flexible and changing allocation of network resources, which, as a consequence, occur more frequently than in physical networks. As a result, virtualized environments regularly exhibit higher levels of traffic variability.

This dynamic behavior might introduce more disorders and anomalies that conventional detection models, which are typically trained on more stable and predictable datasets, may not have seen before and therefore are not prepared to handle. This variability even complicates the learning process on traffic data from virtual networks, as it makes it challenging to establish consistent patterns that are crucial for accurate detection.

Moreover, the additional metadata in GENEVE packets can lead to greater variability in packet size and structure, which might confuse models trained on more uniform, non-virtualized traffic. This variability likely contributes to the lower accuracy observed for GENEVE-encapsulated traffic compared to VXLAN.

6.3. Optimization for Performance

Virtual network protocols are designed to optimize for network performance and scalability rather than for security monitoring and anomaly detection. The optimization techniques include load balancing and dynamic routing. Such features can further obscure the traffic patterns and thus mask malicious activities and benign anomalies alike. If detection systems fail to recognize anomalies in such situations, the effectiveness of the detection system is reduced.

6.4. Findings

The encapsulation overhead, especially with varying additional headers, the increased traffic dynamic, which brings ongoing changes of the virtual network structure, and the performance optimization in virtual network environments that obscure traffic patterns further together result in a reduced detection efficacy observed with virtualized protocols.

While a certain level of reduction seems inevitable and even tolerable, the observed reduction in detection accuracy especially for GENEVE is at a level that necessitates further investigations.

7. Conclusions

Cloud environments have become indispensable in modern networks, offering unparalleled flexibility to both users and providers. These environments leverage various virtualization techniques to create dynamic and adaptable infrastructures. While this flexibility benefits administrators and users, it presents challenges for IT security processes, especially the detection of intrusions and anomalies.

In this paper, we explore a model for intrusion detection in virtual networks. Our model reaches an accuracy of $>$98% when trained with traditional network data, providing a suitable model for detecting outliers. The dynamic nature of these environments with possible changes like virtual machine (VM) migrations can confuse DL-based intrusion detection, leading to false positives or false negatives. Our contribution, and thus the novelty of our research lies in quantifying the extent to which virtual networks can confuse DL-based intrusion detection trained with traditional network data.

Our model is characterized by a high precision and low recall, which indicates a specific trade-off in its predictive capability. The high precision, reflecting a low rate of false

positives, suggests that the model is adept at correctly identifying true positive instances with minimal errors. This is particularly profitable in scenarios where the cost of false positives is high, ensuring that when a positive prediction is made, it is highly likely to be accurate. However, the concomitantly low recall denotes a significant number of false negatives, implying that the model fails to detect a substantial proportion of actual positive cases. This limitation indicates a trade-off with a low detection rate of positives (based on the recall), but, if a positive detection is made, the flow is malicious with a high probability.

In virtualized environments, where traffic patterns are more dynamic and complex, the decrease in accuracy for encapsulation protocols might be critical. This significant drop suggests that the model struggles to adapt to the additional layers of encapsulation and the resulting changes in traffic characteristics. In practice, this could mean that the IDS may miss a substantial number of intrusions or incorrectly classify benign traffic as malicious, leading to operational challenges such as false alarms or undetected attacks.

These misclassified data might result in irregularities in the further processing of network traffic, which has implications for providers of virtual networks. To address this issue, it is important for the provider to refine the deployed algorithms and models, taking into account potential variations, which creates an application scenario for applying our results. Consequently, periodic redefinition or purification of network traffic is essential to mitigate these challenges.

Future work will comprise an analysis of the reasons for the low accuracy and recall values to gain deeper insights into possible improvements, to compare this model's performance with a model trained with specifically crafted network data comprising virtual network traffic, as well as comparing the model's performance with other deep learning systems and with rule-based systems (either crafted by experts or generated as decision trees from labeled network data). Moreover, more detailed and refined models might be explored that also incorporate latency and bandwidth consequences from encapsulation overhead.

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

Funding: This research received no external funding.

Data Availability Statement: The data are available upon request from the authors.

Acknowledgments: We thank the editor and the anonymous reviewers for their constructive comments that helped to improve our article.

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

References

1. Spiekermann, D.; Eggendorfer, T. Challenges of network forensic investigation in virtual networks. *J. Cyber Secur. Mobil.* **2016**, *5*, 15–46. [CrossRef]
2. Egho-Promise, E.; Idahosa, S.; Asante, G.; Okungbowa, A. Digital Forensic Investigation Standards in Cloud Computing. *Univ. J. Comput. Sci. Commun.* **2024**, *3*, 23–45. [CrossRef]
3. Hussain, J.; Hnamte, V. Deep Learning Based Intrusion Detection System: Software Defined Network. In Proceedings of the 2021 Asian Conference on Innovation in Technology (ASIANCON), Pune, India, 27–29 August 2021; pp. 1–6. [CrossRef]
4. Yu, Y.; Long, J.; Cai, Z. Session-Based Network Intrusion Detection Using a Deep Learning Architecture. In Proceedings of the Modeling Decisions for Artificial Intelligence, Kitakyushu, Japan, 18–20 October 2017. [CrossRef]
5. Kocher, G.; Kumar, G. Machine learning and deep learning methods for intrusion detection systems: Recent developments and challenges. *Soft Comput.* **2021**, *25*, 9731–9763. [CrossRef]
6. Sajassi, A.; Drake, J.; Bitar, N.; Shekhar, R.; Uttaro, J.; Henderickx, W. *Network Virtualization Overlay Solution Using Ethernet VPN (EVPN). RFC8365*; RFC Editor: Fremont, CA, USA, 2018. [CrossRef]
7. ETSI. CYBER. Middlebox Security Protocol. Part 3: Enterprise Transport Security. 2019. Available online: https://www.etsi.org/deliver/etsi_ts/103500_103599/10352303/01.02.01_60/ts_10352303v010201p.pdf (accessed on 5 September 2024).
8. CVE-2019-9191. 2019. Available online: https://nvd.nist.gov/vuln/detail/CVE-2019-9191 (accessed on 5 September 2024).

9. Hoffman-Andrews, J. ETS Isn't TLS and You Shouldn't Use It. 2019. Available online: https://www.eff.org/deeplinks/2019/02/ets-isnt-tls-and-you-shouldnt-use-it (accessed on 5 September 2024).
10. Koch, R. Systemarchitektur zur Ein- und Ausbruchserkennung in Verschlüsselten Umgebungen. Ph.D. Thesis, Universität der Bundeswehr München, Neubiberg, Germany, 2011.
11. Ho, C.Y.; Lai, Y.C.; Chen, I.W.; Wang, F.Y.; Tai, W.H. Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems. *IEEE Commun. Mag.* **2012**, *50*, 146–154. [CrossRef]
12. Samrin, R.; Vasumathi, D. Review on anomaly based network intrusion detection system. In Proceedings of the 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, India, 15–16 December 2017; pp. 141–147. [CrossRef]
13. Spiekermann, D.; Keller, J. Challenges of Network Traffic Classification Using Deep Learning in Virtual Networks. In Proceedings of the International Workshop On Digital Forensics (IWDF), Hamburg, Germany, 26–30 September 2022; Gesellschaft für Informatik: Bonn, Germany, 2022; pp. 99–108. [CrossRef]
14. Russell, S.; Norvig, P. *Artificial Intelligence, Global Edition*, 4th ed.; Pearson: London, UK, 2021; p. 1168.
15. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
16. Abbasi, M.; Shahraki, A.; Taherkordi, A. Deep learning for network traffic monitoring and analysis (NTMA): A survey. *Comput. Commun.* **2021**, *170*, 19–41. [CrossRef]
17. Krupski, J.; Graniszewski, W.; Iwanowski, M. Data transformation schemes for cnn-based network traffic analysis: A survey. *Electronics* **2021**, *10*, 2042. [CrossRef]
18. Rajesh, P.; Ismail, B.M.; Alam, M.; Tahernezhadi, M.; Agrawal, M. Network Forensics Investigation in Virtual Data Centers Using ELK. In Proceedings of the 2021 International Symposium on Electrical, Electronics and Information Engineering, Seoul, Republic of Korea, 19–21 February 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 175–179. [CrossRef]
19. Wang, Z.; Yang, M.; Li, S. Study of Network-based Intrusion Detection System for virtualization. In Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 16–18 April 2010; Volume 6, pp. 595–598. [CrossRef]
20. Modi, C.N.; Patel, D.R. A feasible approach to intrusion detection in virtual network layer of Cloud computing. *Sādhanā* **2018**, *43*, 114. [CrossRef]
21. Chung, C.J.; Khatkar, P.; Xing, T.; Lee, J.; Huang, D. NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems. *IEEE Trans. Dependable Secur. Comput.* **2013**, *10*, 198–211. [CrossRef]
22. Dina, A.S.; Manivannan, D. Intrusion detection based on Machine Learning techniques in computer networks. *Internet Things* **2021**, *16*, 100462. [CrossRef]
23. Zhang, H.; Wang, W.; Huang, L.; Wu, J.; Zhang, F.; Shi, K. Correlating Intrusion Detection with Attack Graph on Virtual Computer Networkings. In Proceedings of the 9th International Conference on Information Systems Security and Privacy, ICISSP 2023, Lisbon, Portugal, 22–24 February 2023; Mori, P., Lenzini, G., Furnell, S., Eds.; SciTePress: Setúbal, Portugal, 2023; pp. 651–658. [CrossRef]
24. Zamrii, I.; Vyshnivskyi, V.; Sobchuk, V. Method of Ensuring the Functional Stability of the Information System based on Detection of Intrusions and Reconfiguration of Virtual Networks. In Proceedings of the Workshop Cybersecurity Providing in Information and Telecommunication Systems (CPITS 2024), Kyiv, Ukraine, 28 February 2024; Sokolov, V., Ustimenko, V., Radivilova, T., Nazarkevych, M., Eds.; CEUR Workshop Proceedings; CEUR-WS.org: Aachen, Germany, 2024; Volume 3654, pp. 252–264.
25. Srinivas, B.V.; Mandal, I.; Keshavarao, S. Virtual Machine Migration-Based Intrusion Detection System in Cloud Environment Using Deep Recurrent Neural Network. *Cybern. Syst.* **2024**, *55*, 450–470. [CrossRef]
26. Hardegen, C.; Pfülb, B.; Rieger, S.; Gepperth, A. Predicting network flow characteristics using deep learning and real-world network traffic. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2662–2676. [CrossRef]
27. Hu, B.; Kamiya, K.; Takahashi, K.; Nakao, A. Piper: A unified machine learning pipeline for internet-scale traffic analysis. In Proceedings of the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
28. 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]
29. Spiekermann, D.; Keller, J. Encapcap: Transforming Network Traces to Virtual Networks. In Proceedings of the 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 28 June–2 July 2021; pp. 437–442. [CrossRef]
30. Nakashima, M.; Sim, A.; Kim, Y.; Kim, J.; Kim, J. Automated feature selection for anomaly detection in network traffic data. *ACM Trans. Manag. Inf. Syst. (TMIS)* **2021**, *12*, 1–28. [CrossRef]
31. Pasyuk, A.; Semenov, E.; Tyuhtyaev, D. Feature selection in the classification of network traffic flows. In Proceedings of the 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), Vladivostok, Russia, 1–4 October 2019; pp. 1–5. [CrossRef]
32. Chovanec, M.; Hasin, M.; Havrilla, M.; Chovancová, E. Detection of HTTP DDoS Attacks Using NFStream and TensorFlow. *Appl. Sci.* **2023**, *13*, 6671. [CrossRef]
33. Bozkır, R.; Cicioğlu, M.; Çalhan, A.; Toğay, C. A new platform for machine-learning-based network traffic classification. *Comput. Commun.* **2023**, *208*, 1–14. [CrossRef]

34. Jiang, L.; Zhang, H.; Cai, Z. A Novel Bayes Model: Hidden Naive Bayes. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1361–1371. [CrossRef]
35. Chowdhury, M.N.; Ferens, K.; Ferens, M. Network intrusion detection using machine learning. In Proceedings of the International Conference on Security and Management (SAM), Las Vegas, NV, USA, 25–28 July 2016; p. 30.
36. Gómez, J.; Riaño, V.H.; Ramirez-Gonzalez, G. Traffic classification in IP networks through Machine Learning techniques in final systems. *IEEE Access* **2023**, *11*, 44932–44940. [CrossRef]
37. Grigoriadou, S.; Radoglou-Grammatikis, P.I.; Sarigiannidis, P.G.; Makris, I.; Lagkas, T.; Argyriou, V.; Lytos, A.; Fountoukidis, E. Hunting IoT Cyberattacks With AI-Powered Intrusion Detection. In Proceedings of the IEEE International Conference on Cyber Security and Resilience, CSR 2023, Venice, Italy, 31 July–2 August 2023; pp. 142–147. [CrossRef]
38. Gross, J.; Ganga, I.; Sridhar, T. *Geneve: Generic Network Virtualization Encapsulation. RFC 8926;* RFC Editor: Fremont, CA, USA, 2020. [CrossRef]
39. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
40. Maciá-Fernández, G.; Camacho, J.; Magán-Carrión, R.; García-Teodoro, P.; Therón, R. UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* **2018**, *73*, 411–424. [CrossRef]
41. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8. [CrossRef]
42. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994. [CrossRef]
43. 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 2018), Madeira, Portugal, 22–24 January 2018; Volume 1, pp. 108–116. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.