Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 13. Issue.01, PP. 61-68, June 2025
https://doi.org/10.22555/pjets.v13i1.1257
E-ISSN 2224-2333
P-ISSN 2222-9930

Dense Neural Network Classification Model for Software Defined Network for Fine-Grained Traffic Routing and Flow Analysis

Khaliq Ahmad[1*], Maheen Danish[2], Asif Raza[3]
[1]Department of Computer Science, Iqra University, Karachi, Pakistan
[2]Department of Computer Engineering, Sir Syed University of Engineering and Technology, Karachi, Pakistan
[3]Department of Computer Science & IT, Sir Syed University of Engineering and Technology, Karachi, Pakistan
**Corresponding author: kkhanzada@hotmail.com*

Abstract:

Traffic classification in SDN environments acts as the centerpiece, greatly enhancing network management, security, and overall performance. In essence, the classification of traffic in the SDN environment involves classifying network traffic into specific classes according to certain criteria type, source, or destination. Through good classification of network traffic, network operators can optimize resources with unprecedented efficiency, leverage better routing, and grant more priority to the essential data flows that improve QoS. This study utilizes an SDN dataset from Kaggle and evaluates the performance of a state-of-the-art classification model. The paper introduces a further improved Deep Dense Neural Network (DDNN) model, optimized with the Adam optimizer, having a remarkable classification accuracy of 90%. In this paper, the Adam optimizer has been adopted because it allows for an adaptive learning rate that improves convergence and stability during training. Besides, this model showed high scores for other metrics: Precision, Recall, and F1-score all exceeded 90%, reflecting the model's ability to classify reliably and with balance. The low network loss indicates that the model not only gives an accurate result but is also consistent in its performance, with very few misclassifications. Fine-grained traffic classification cannot be underestimated in SDN, given the wide gamut that runs applications over SDN networks. The DDNN model handles diverse types of traffic with a lot of efficiency through extracting deep features from complex datasets, hence increasing the capability of SDN to meet the ever-evolving demands in networks.

Keywords: *SDN, security, protection, network, threat, attack.*

I. INTRODUCTION

Modern networks in the twenty-first century provide significant flexibility to both businesses and individual users, but this comes with increased complexity. Managing and controlling these networks has become a highly intricate and specialized task, making legacy networks challenging to automate [1, 2]. In this scenario, Software Defined Networking (SDN) has emerged as a promising solution with the potential to transform the networking landscape. SDN allows network administrators to gain enhanced control over traffic flows while simplifying the programming and modification of network policies to suit user requirements. This is achieved by shifting control and policy functions from numerous distributed devices to one or more centralized general-purpose servers [3]. In recent years, interest in SDN research and implementation has grown significantly among academia, industry professionals, and network operators. Although the foundational concepts behind SDN have been around for over two decades [4], the developments closely associated with SDN are comparatively recent.

Due to its programmable nature, Software Defined Networks (SDNs) are vulnerable to various forms of malicious software and attacks. The abstraction of available traffic flows and underlying hardware resources at the SDN controller facilitates intelligence gathering from existing resources, but this can also be exploited for attacks, manipula-

tions, or even complete network reprogramming. Similarly, the southbound interface of SDN is particularly susceptible to diverse threats, including denial of service and side-channel attacks. Furthermore, configuration errors in SDNs can have more severe impacts compared to traditional networks. SDN agents are also potential targets for injecting false flows. Given the features and architecture of SDNs, cyberattacks initiated through them can result in more extensive and devastating consequences than those involving conventional networks [5].

II. LITERATURE REVIEW

Software Defined Networking (SDN) was created to drive innovation and simplify programmatic control of network data paths. As illustrated in Figure 1, separating forwarding hardware from control logic enables the easier deployment of new protocols and applications, simplified network visualization and management, and the consolidation of various middlebox functions into software-based control. Instead of distributing policy enforcement and protocol operations across numerous devices, the network is streamlined into basic forwarding hardware and a centralized decision-making network controller [6].

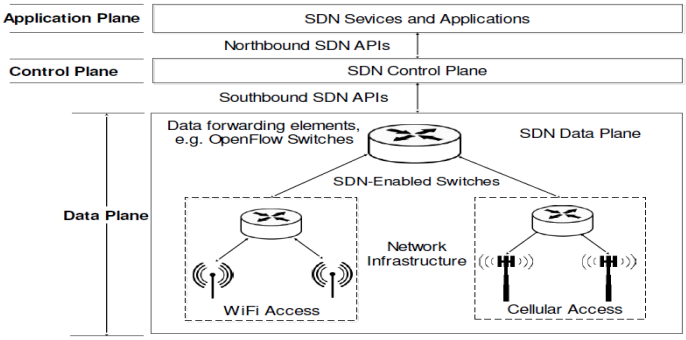

Figure 1: The SDN Architecture

OpenFlow: Driven by the SDN principle of decoupling the control and data forwarding planes, [6], like ForCES, standardizes information exchange between the two planes. In the OpenFlow architecture, illustrated in Figure 1, the forwarding device, or OpenFlow switch, contains one or more flow tables and an abstraction layer that securely communicates with a controller via the OpenFlow protocol.

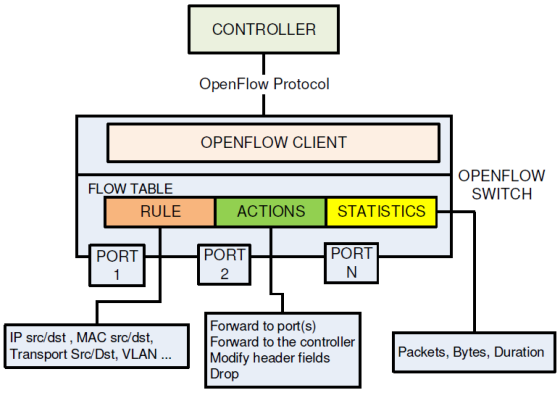

Figure 2: OpenFlow Architecture

As shown in Figure 2, operating system alteration involves the damage or modification of components or the entire operating system of SDN elements, such as controller or forwarder nodes. Unauthorized applications can harm the three higher layers, while malicious applications can target all corresponding layers, posing a persistent challenge for SDNs [5].As the foundation of the environment, ensuring SDN agent security is crucial. An attacker could potentially access the SDN controller by exploiting one or more vulnerable agents within the network [7].

III. RESEARCH METHODOLOGY AND DESIGN

A Distributed Denial of Service (DDoS) attack targets the availability of network resources by overwhelming them. This is achieved through a coordinated effort involving multiple devices, which may participate knowingly or unknowingly in the attack. The attacker floods network resources with excessive, non-essential traffic, leading to exhaustion of resources. Consequently, malicious traffic is served while legitimate packets are delayed or dropped due to congestion or packet overflow. Research highlights that DDoS attacks are among the simplest and most prevalent threats to network security [8]. Figure 5 presents a classification of key DDoS attack types.

In 2012, Radware introduced the DefensePro attack mitigation solution [9], a comprehensive DDoS protection system comprising DefenseFlow and DefensePro modules. Its Anti-DoS module monitors incoming traffic to the controller and compiles statistical data. Many enterprises have adopted sFlow, a technology proposed by InMon Technologies in 2002 [10, 11], as a powerful tool for traffic analysis in SDN networks. In 2014, a solution combining sFlow and OpenFlow was proposed [12] to enable scalable anomaly detection. Another OpenFlow-based DDoS mitigation scheme was introduced in 2014 [13], leveraging traffic statistics to identify attacks. Additionally, Of-Guard [14] utilizes traffic statistics for mitigation, while Avant-Guard [15], proposed in 2013, detects and mitigates DDoS attacks by analyzing TCP connection states to distinguish between legitimate and malicious traffic. In 2015, DaMask [16, 17] introduced a network virtualization-based approach to combat DDoS attacks.

A. Deep learning-based Detection:

Deep Dense Neural Networks (DDNNs) are a class of neural networks characterized by their deep, densely connected architecture. These networks are adept at learning complex patterns and representations from data, making them highly effective for applications such as image recognition and natural language processing. Deep learning-based methods enable networks to make decisions autonomously, without external intervention. While various machine learning techniques have been employed for detecting and mitigating attacks in traditional networks, their application in securing SDN remains limited. Nevertheless, these approaches show significant potential for mitigating DDoS attacks in SDN environments [18-20].

B. Traffic Classes:

This bar chart displays the quantity of data points across various traffic classes in a network. Each bar represents a different traffic category (e.g., chat, streaming, audio, file transfer), with the height of each bar corresponding to the quantity of traffic in that category. "FILE-SKYPE" has the highest quantity, followed by categories like "AUDIO-HANGOUTS" and "EMAIL." Lower quantities are seen in categories like "CHAT-HANGOUTS" and "STR-VIMEO," indicating a distribution of traffic volume across different applications and services, as shown in Figure 3.

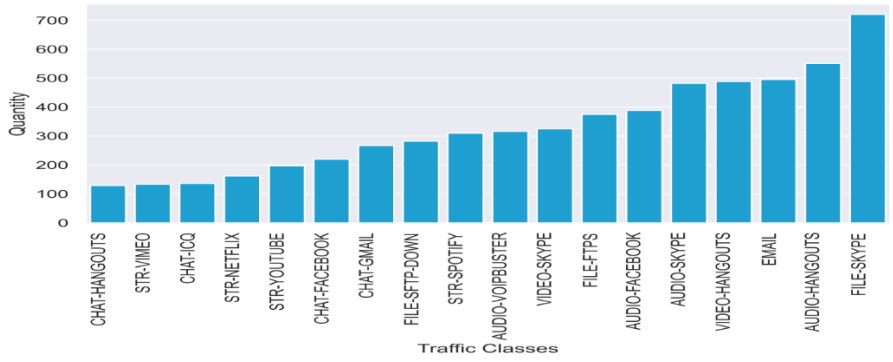

Figure 3: Traffic Classes

C. Cross Validation:

Dense Neural Network architecture uses the loss function Cross Entropy Loss and configures the Adam optimizer with a specified learning rate and weight decay for regularization. Using K-Fold cross-validation (`KFold`) with `k` splits, the code initializes arrays to track evaluation metrics.

IV. RESULTS AND DISCUSSION

The graph shows a series of line graphs comparing training and testing accuracy across five different K-Fold cross-validation splits (KFold 1 to KFold 5) over 50 epochs. In each graph, the training accuracy consistently rises higher than the testing accuracy, indicating that the model performs better on the training data than on the test data. This pattern suggests potential over-fitting, as training accuracy improves more significantly than testing accuracy across all folds, as shown in Figure 4.

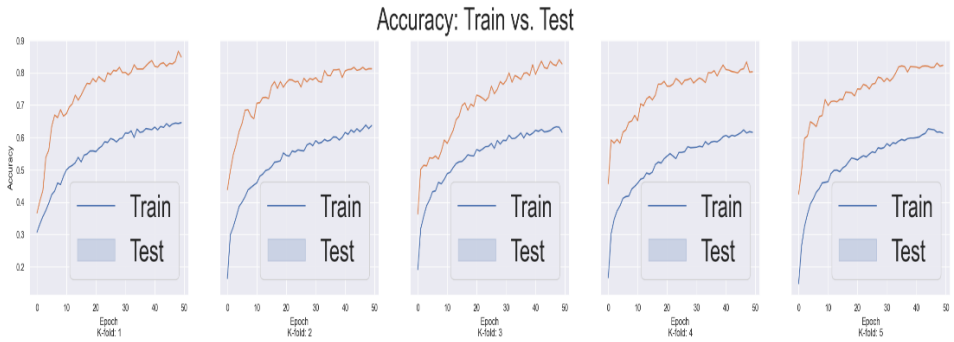

Figure 4: Accuracy

A. Training and Testing Loss:

The training loss consistently remains lower than the testing loss in each fold, as shown in Figure 5, suggesting that the model fits the training data better than the testing data.

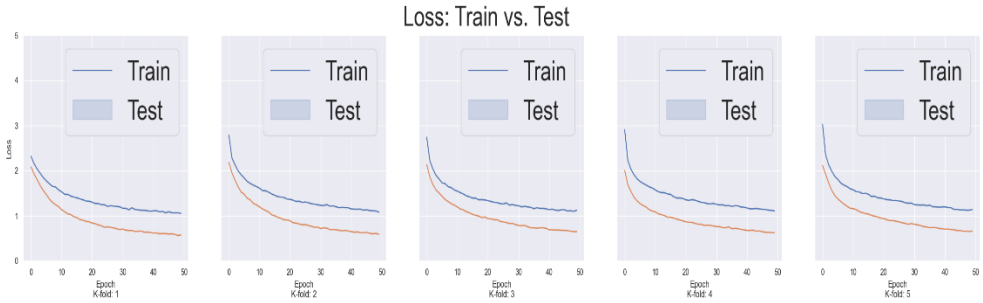

Figure 5: Loss

B. Confusion Matrix for Cross-validation:

Darker diagonal squares signify correct classifications (i.e., where the predicted label matches the actual label), while off-diagonal values indicate misclassifications. The overall trend of darker diagonal lines suggests that the model generally performs well, with most predictions aligning with the actual labels, as shown in Figure 6.

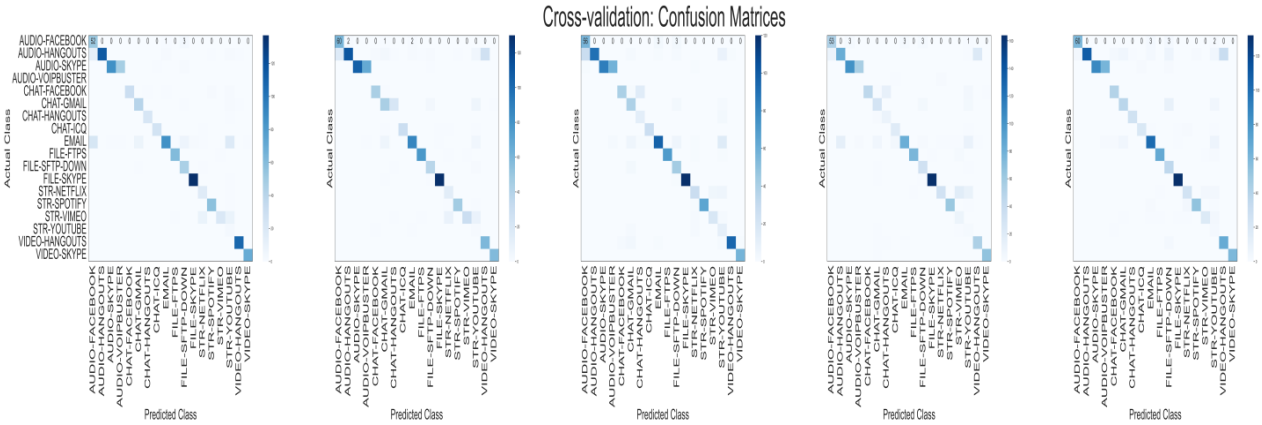

Figure 6: Cross Validation: Confusion Matrix

Table 1 presents performance metrics for a classification model evaluated across five cross-validation folds. Accuracy, which measures the proportion of correct predictions, ranges from 80.30% to 84.90%, indicating generally strong overall performance. Precision, which assesses the accuracy of positive predictions, varies between 72.99% and 80.77%, while recall, which captures the proportion of actual positives correctly identified, ranges from 74.54% to 83.53%. The F1-score, a harmonic mean of precision and recall, is consistently high, ranging between 0.9887 and 0.9914, suggesting that the model maintains a solid balance between precision and recall despite fluctuations in accuracy. The consistently high F1-scores imply robust classification performance, even though precision shows more variability, indicating that the model slightly favors recall over precision, meaning it's better at capturing positive instances than avoiding false positives.

Table 1: Metrics

Fold	Acc.	Precision	Recall	F1-s
1	0.84896	0.80774	0.83528	0.99138
2	0.8125	0.74522	0.74541	0.98932
3	0.82726	0.78657	0.79998	0.99016
4	0.80295	0.72987	0.7775	0.98874
5	0.82292	0.78951	0.80109	0.98987

C. Evaluation of K-Fold Cross Validation:

The chart shows that accuracy is consistently the highest across all folds, with scores above 0.8, indicating good overall model performance. Precision, recall, and F1-score are slightly lower but remain close to each other in each fold, suggesting balanced performance in terms of correctly identifying true positives and minimizing false positives. The metrics do not vary significantly between the folds, implying that the model's performance is relatively stable across different subsets of the data, as shown in Figure 7.

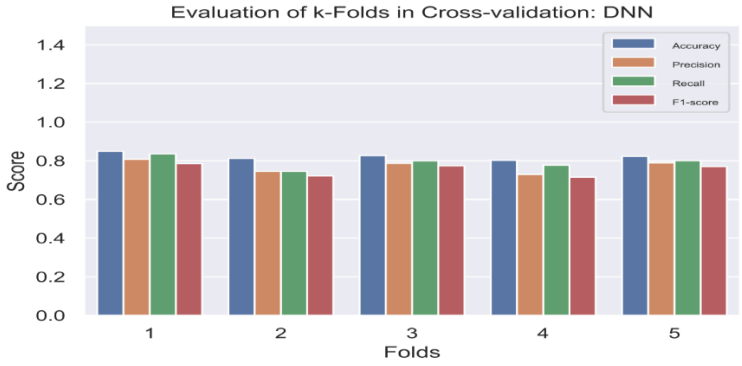

Figure 7: K-Fold Evaluation

D. Train vs. Test Accuracy:

The graph consists of multiple line charts illustrating the performance metrics (Accuracy, Precision, Recall, and F1-score) over training epochs across five different cross-validation folds (K=1 to K=5) as mentioned in Figure 8. Each chart shows how these metrics improve as training progresses. Accuracy (blue line) is consistently the highest metric, quickly reaching a plateau close to 1.0, indicating that the model achieves high correctness early on. Precision, recall, and F1-score gradually increase over the epochs, suggesting that the model improves its ability to correctly classify positive instances as it learns. The metrics appear to stabilize after around 20-30 epochs, demonstrating that further training yields diminishing returns. The trends are consistent across all folds, indicating that the model's training process is stable and reliable.

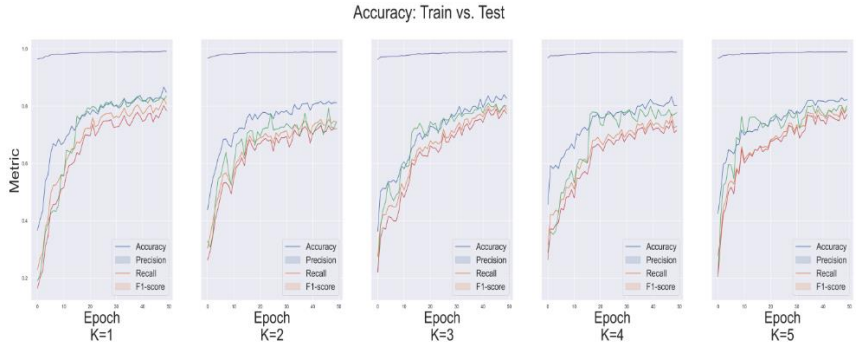

Figure 8: Train vs Test Accuracy

E. ROC /AUC Graph:

Multi-class ROC curve shows error deflection per class as mentioned in Figure 9. The x-axis represents the false positive rate (1 - specificity), while the y-axis shows the true positive rate (sensitivity). Each colored line corresponds to a different class, with the area under the curve (AUC) indicated in the legend. An AUC closer to 1.0 signifies that the model performs well in distinguishing that class from others, with 1.0 representing a perfect classifier. Most classes here have AUC values near 1.0, indicating high accuracy. For example, "Class FILE-SKYPE" achieves an AUC of 1.000000, implying perfect classification, while other classes like "Class CHAT-HANGOUTS" also perform well, with slightly lower AUC scores. This demonstrates that the model has excellent performance across multiple classes, as shown by the high sensitivity and low false positive rates.

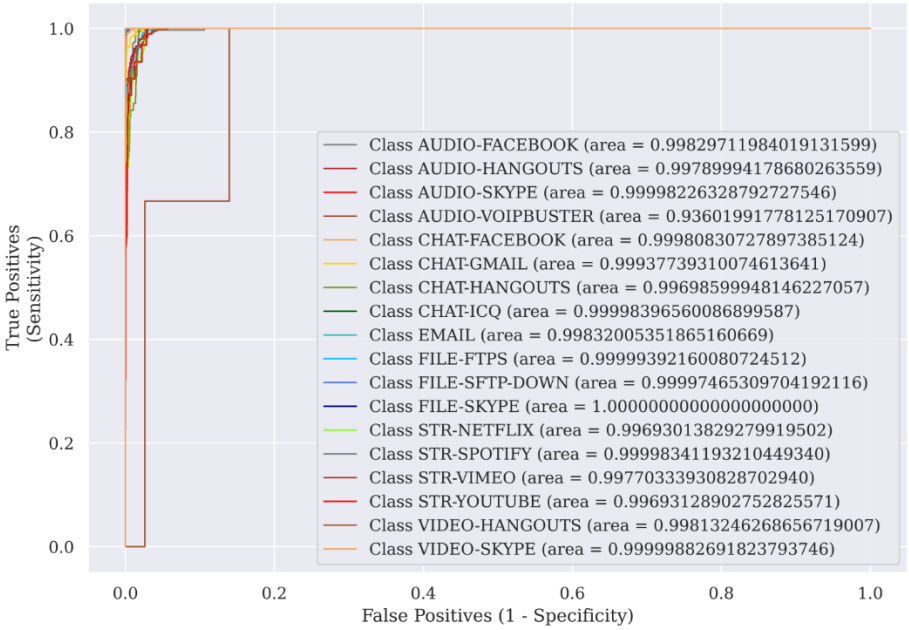

Figure 9: ROC /AUC

F. Accuracy Line Plot:

This line plot illustrates the performance metrics of a deep learning model. The y-axis represents various metrics (Accuracy, Precision, Recall, and F1-score), while the x-axis shows the number of epochs. The Accuracy initially rises sharply and plateaus around 0.85, indicating the model's improved classification ability over time. The other metrics follow a similar trend, steadily increasing and stabilizing between 0.75 and 0.85. This suggests that the model's predictions are not only accurate but also balanced in terms of precision and recall. Overall, the graph demonstrates that the model converges effectively, achieving consistent and robust performance across all metrics after about 20-30 epochs. As shown in Figure 10, this pattern suggests successful model training with minimal overfitting.

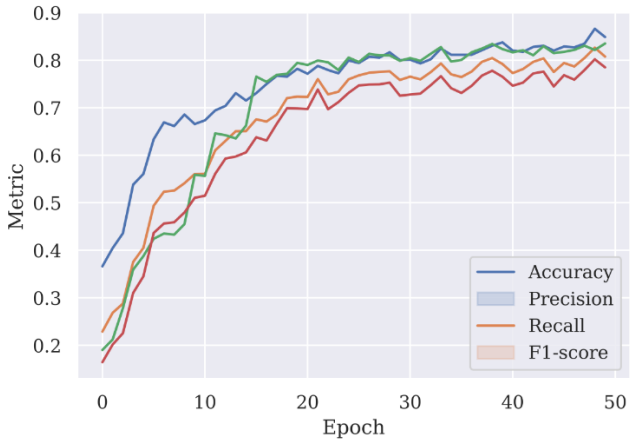

Figure 10: Accuracy

V. CONCLUSION

Traffic classification plays a central role in SDN, significantly enhancing network management, security, and performance. By categorizing network traffic by type, source, or destination, operators can efficiently optimize resources, improve routing, and prioritize critical data flows to enhance QoS. This fine-grained traffic classification boosts SDN's capacity to meet evolving demands, effectively handling diverse traffic through deep feature extraction from complex datasets.

REFERENCES

1. T. Anderson, L. Peterson, S. Shenker, J. Turner, Overcoming the internet impasse through virtualization, Computer 38 (4) (2005) pg: 34–41.
2. HP, Deliver HP Virtual Application Networks, 2012. <http://h17007.www1.hp.com/docs/interopny/4AA4-3872ENW.pdf>.
3. Sixto Ortiz Jr., "Software defined networking The verge of breakthrough", IEEE Computer Society, 2013.
4. Sezer, S. et al, "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks", IEEE Communications Magazine, Volume: 51, Issue: 7, 2013, pp. 36- 43.
5. A. Akhunzada et al, "Securing the Software Defined Networks: Taxonomy, Requirements, and Open Issues", IEEE Communication Magazine, Dec. 2014.
6. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69–74, 2008.
7. A. Akhunzada et al, "Man-At-The-End Attacks: Analysis, Taxonomy, Human Aspects, Motivation and Future Directions", Journal of Network and Computer Applications, 2014.
8. N. Dayal et al, "Research Trends in Security and DDoS in SDN", Security and Communication Networks, Security Comm. Networks 2016.
9. "Denial-of-Service (DoS) Secured Virtual Tenant Networks (VTN)", Whitepaper by Radware and NEC Corporation, 2012.

10. Reves J, Panchen S., “Traffic Monitoring with Packetbased Sampling for Defense Against Security Threats”, Whitepaper, 2002.
11. “Traffic Monitoring using sFlow, as sFlow”, Whitepaper. by InMon technologies, 2003.
12. Giotis K. et al, “Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”, Computer Networks 2014; vol. 62, pp. 122–136.
13. Dillon C, Berkelaar M., “OpenFlow DDoS Mitigation”, Technical report, 2014, <http://www.delaat.net/rp/2013-2014/p42/report.pdf;2014Feb9>
14. Wang H, Xu L, Gu G., “OF-GUARD: a DoS attack prevention extension in software-defined networks”, Open Network Summit, USENIX, 2014.
15. Shin S, Yegneswaran V, Porras P, Gu G., “AVANTGUARD: scalable and vigilant switch flow management in software-defined networks”, Proceedings of Conference on Computer and Communication Security (CCS), ACM, 2013; pp. 413–424.
16. Shin S, Guofei G., “Attacking Software Defined Networks: A First Feasibility Study”, HotSDN, 2013.
17. Wang B, Zheng Y, Lou W, Thomas Hou Y., “DDoS attack protection in the era of cloud computing and software-defined networking”, Proceedings of International Conference on Network Protocol (ICNP), IEEE, 2014; pp. 624–629.
18. Ashraf J, Latif S., “Handling intrusion and DDoS attacks in software defined networks using machine learning techniques”, Proceedings of National Software Engineering Conference, 2014; pp. 55–60.
19. Braga R, Mota E. Alexandre Passito, “Lightweight DDoS Flooding Attack Detection using NOX/OpenFlow”, Proceedings of LCN, 2010, pp. 408–415.
20. Kokila RT, Selvi ST, Govindarajan K., “DDoS Detection and Analysis in SDN-based Environment Using Support Vector Machine Classifier”, Computer Networks, 2014.