International Journal of Engineering Sciences & Emerging Technologies, May 2024.
ISSN: 22316604 Volume 11, Issue 3, pp: 501-509 ©IJESET

MACHINE LEARNING TECHNIQUES AND ITS APPLICATION IN SOFTWARE DEFINED NETWORK

VikasVerma [1], Hemant Mathur [2]
[1,2] Department of Computer Science and Engineering,
Jaipur National University, Jaipur, India
[1]vikasverma@jnujaipur.ac.in
[2]hemantmathur@jnujaipur.ac.in

ABSTRACT

The ongoing advancement of systems Network is causing it to turn out to be progressively mind boggling. SDN (Software Design Network) has become a new examination field. The intensity of knowledge can be accomplished through Software Defined Networking (SDN), and the ongoing advancement of Machine Learning (ML), provides a number of strategies to address distinctive system challenges. Routing which additionally fulfills Quality-of-Service (QoS) is one of the significant difficulties in SDN based systems, it is likewise run of the mill when there are various kinds of streams that exist in a similar system. Software-Defined Networks (SDN) changes the conveyed and equipment driven inheritance to organize into a coordinated and dynamic system that gives an exhaustive arrangement the mix of the ML calculation. The system wide information gave by SDN can be utilized for productive traffic routing in the system. In this work, we investigate and Analysis the appropriateness of AI calculations for choosing the least clogged course for routing traffic in a SDN empowered system.

KEYWORDS

Software-defined network, Machine Learning, Routing Optimization

1. INTRODUCTION

Software-Defined Networking is a systems administration worldview that empowers unified command over the system foundation by isolating control and information planes. It gives a programmable interface between arrange gadgets and its brought together controller.

The ongoing appearance in versatile mobile communication frameworks has energized Software Defined Networking (SDN) with the mix of programming characterized innovation and the Internet, a Software-Defined Network SDN makes another and alluring answer for arranging advancement. In the SDN design architecture network device control plane is isolated from the information sending data plane. The logical device (SDN-Controller) of the control plane screens and acquires the present status of the system continuously. It empowers organize supervisors to control arrange traffic sending as indicated by their characterized directing guidelines.

All the components of the SDN are efficiently taken in control by the control plane, which really fills in as the Networking Operating System. This brought together controller is powerfully programmable. In addition, the controller gathers organize states, bundles, and data about stream in the system. The controller has a general perspective on the system. We present here brief foundation information on Software-characterized systems from the edge of its engineering and stream.

(i). Architecture of Software defined networking

SDN has gained popularity and attention around the world in recent years. The Open Networking Foundation (ONF) [1] is a non-profit organization specializing in the development and standardization of software-defined networking. According to ONF, SDN can be defined as: “The SDN architecture separates the control and data planes, and have centralizes network intelligence and state, and abstracts the underlie network infrastructure from the application.” [2].

On the basis of above definition, the architecture of a software-defined network consisting of three layers, including the data plane, control plane, and application plane, is presented. Figure 1 shows the architectural components of each phase and their interactions. Figure 1 also consists of a detailed information of these three components.

a) Data Plane: In the SDN architecture data plane is the bottom layer of SDN engineering now and then it is otherwise called infrastructure plane. This plane has physical sending devices like switches there may also virtual switches. A virtual switch is only a program act as a switch, which can run on basic working frameworks, for example, Windows or Linux.

Open vSwitch[3], Indigo[4], and Pantou[5] are pre-defined the implementations of programmed virtual switch. Unlike virtual switch physical switches are hardware-based switches. Switches are of two sorts one is physical switches which are actualized on open system equipment (e.g., NetFPGA[6]) and the other one is executed on systems administration equipment dealer switches Switchblade [7] and Server Switch [8] are two NetFPGA-based physical switches. Nowadays, some systems administration equipment sellers, for example, HP, NEC, Huawei, Juniper, and Cisco, have likewise empowered SDN conventions in their own trader switches. Propositions Virtual switches bolster total highlights of SDN conventions, while physical switches have come up short on the adaptability and highlight culmination. Be that as it may, physical switches have a larger stream sending rate in contrast with virtual switches.

b) Control Plane: The control plane is fundamentally the "Heart" of Software Defined Network frameworks, which can control all system assets utilizing a program, update sending rules progressively dependent on organize state, and make arrange organization adaptable and nimble. The main part of the Control Plane is the programmable controller of SDN, which controls the controls the correspondence between sending devices and application.

On the one hand, the controller detects state data and abstracts it from the information layer to the application layer. The controller changes over the prerequisites from applications into various approaches and programs and gives the sending devices. Moreover, the controller gives basic functionalities that the entirety of the system applications need, for example, the most optimized way of routing, organize geography stockpiling, device design, and state data notifications, and so forth. There are number of controller, like NOX [9], POX[10], Floodlight[11], Ryu[12], OpenDayLight[13], and Beacon[14].

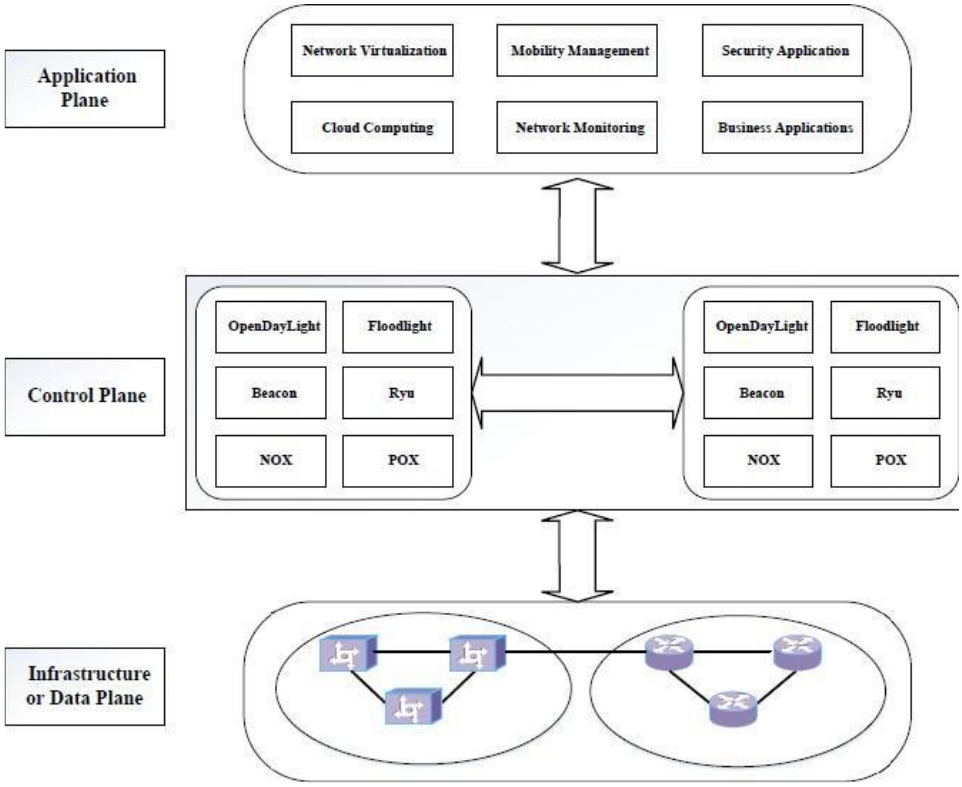

Fig 1: The layer architecture of SDN

c) Application Plane: The top layer of the design of a SDN is the application plane, which is essentially join with business applications. Business Management and various administrations can be accomplished through these applications. The controllers' NBIs give all data about the system state. In light of this data and necessities of a specific business application, the applications can make the control rationale to change arrange practices.

2. MACHINE LEARNING

Machine learning is only a group of methods in Artificial Intelligence regions and has been utilized for huge scope for information mining, and for various issues that permit gaining from preparing models and foreseeing for new highlights and data.

There are two main phases of each machine learning algorithm: training phase and decision-making phase as shown in the Fig. 2.

In the training phase, machine learning algorithm we provide some training example on the basis of these training example our model is learn facts and figure inside the data. In the decision taking phase, our experienced model can predict value for each new given input.

There are basically four Types of Machine learning algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning, these are shown in Fig. 3.

Fig 2:

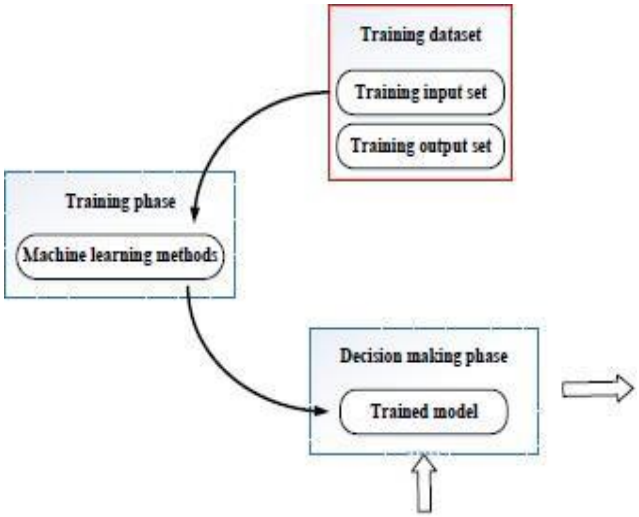

Fig 2: The Function of each phase of Machine learning Algorithm

a) Supervised Learning

In supervised learning, the algorithm is gaining from two kinds of datasets, a training dataset, and a test dataset. The supervised learning algorithm is used as a labeled training dataset (that is, inputs and known outputs). If ambiguous information is passed to the framework after training, the model can predict the output. Therefore, the learner's main purpose is to develop rules or programs that can classify new examples (from a given test set) by analyzing a given example to obtain the expected results [16].] [17.].

b) Unsupervised Learning

Despite of supervised learning, Unsupervised learning algorithm take input the dataset which doesn't have marks (i.e., there is no level). Essentially, a unsupervised learning algorithm is basically for discovering structures, information and example in a unlabeled dataset by combining the information into different gatherings dependent on the closeness in the highlights. The unsupervised learning algorithms are commonly utilized in bunching and information affiliation [15] [17]. Broadly unsupervised learning algorithms are like DBscan and self-arranging maps.

c) Semi-supervised Learning

Semi-supervised learning belongs to the algorithms [18], [19] used for both labeled and unlabeled datasets. Semi-supervised learning is widespread for two reasons. For one, real-world applications primarily process labeled data, which is very expensive and its difficult, but processing a large amounts of unlabeled data is relatively very easy and inexpensive. Secondly, the efficient making use of unlabeled data during the training process reflects an improvement in the execution of the prepared model. To sure about enhance employments of unlabeled information, presumptions ought to be seen in semi-regulated learning, for example, perfection suspicion, bunch supposition, low-thickness detachment suspicion, and complex presumption. Pseudo Labeling [20], [21] is a simple and advanced semi-adjustment learning method. The basic pseudo-tagging methodology is to prepare the framework model with the tagged information. It then uses this trained model to predict pseudo-labels for unlabeled data. Finally, we retrain the system model by combining the labeled data with the predicted pseudo-labels. There are other semi-supervised learning algorithms such as: B. EM (Expectation Maximization), SVM and Graph Methods. All of them are based on different assumptions [22]. For example, EM extends group guessing. The SVM expands on the suspicion of thin partitions, whereas the chart strategy works for a different assumption.

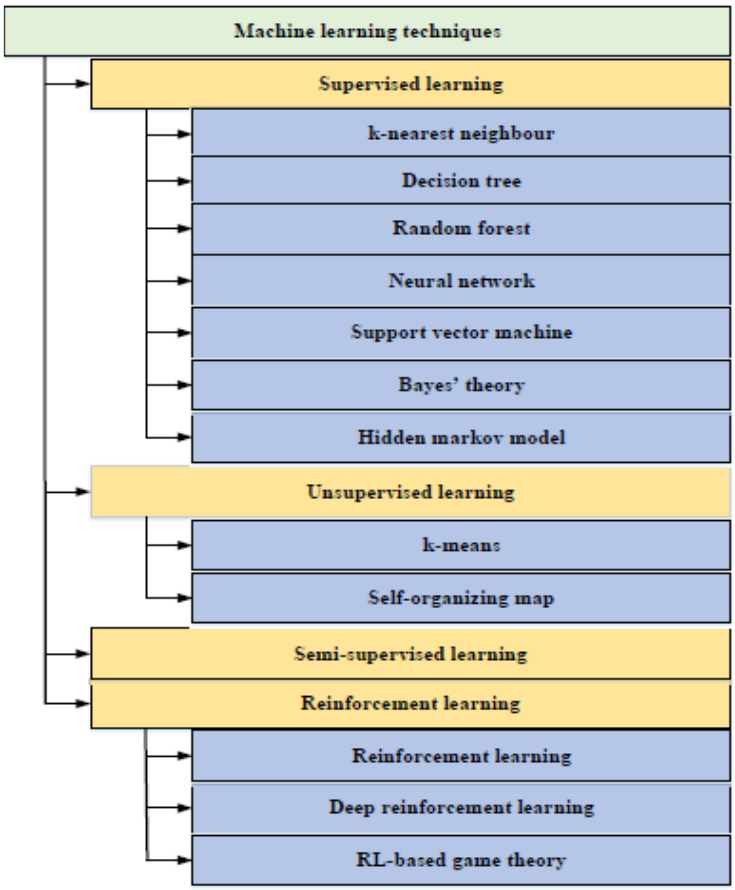

Fig. 3. Common machine learning algorithms applied to SDN.

d) Reinforcement Learning (RL)

One greater famous studying method is Reinforcement Learning [23], [24]. It is broadly used and it makes use of a agent, a state-area S with a movement area A. An agent is a studying frame that interconnects with its surroundings to examine the satisfactory parameter to do so for maximizing its long-time period results. When making use of Reinforcement Learning to the software program described networking, the manipulate aircraft usually works like a agent and the community is the surroundings. Controller constantly observes the community repute and learns from those states to attain the belief for controlling statistics forwarding. A Reinforcement Learning device is proven in Fig. 4.

In particular, Lets for each time t, the operator watched a state st and picks it for activity from the activity space A, which gets a prompt prize rt which demonstrates how fortunate or unfortunate the activity is, and transmit it to the following state that is st+1. The principle inspiration for the specialist is to become familiar with the ideal conduct lets state π strategy which is an immediate planning from the state space S for the activity space A (π: S $\rightarrow$A) to boost the normal long haul reward.

As stated earlier we have a behavior policy , the agent can estimate best action related to a given particular state. In Reinforcement Learning, the worth capacity is utilized for ascertaining the drawn out reward of an activity for a given state.

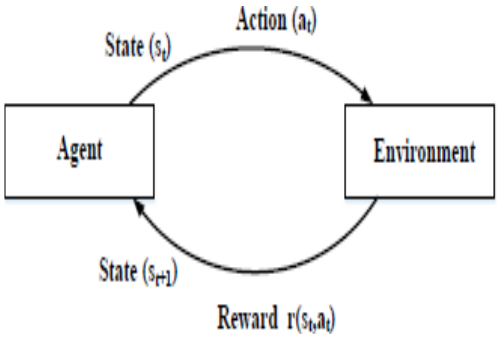

Fig. 4. A Reinforcement Learning system.

e) Deep Reinforcement Learning (DRL)

The main advantage of reinforcement learning is that it works well without prior knowledge of rigorous scientific models of the environment. However, traditional reinforcement learning approaches have some drawbacks. For example, the combination frequency of the ideal action strategy π is low and cannot cope with the problems of multidimensional state space and action space. These shortcomings have already been corrected by Deep Reinforcement Learning (DRL) [26] to [27].An important idea for DRL is to take advantage of the profound NN's amazing capacity estimation characteristics to estimate the capacity of significant values. After preparing a deep neural network, given a pair of state actions as information, the DRL can estimate the reward to be withdrawn given a pair of state activities as data. The estimation results can be controlled by the administrator to select the best activity.

3. APPLICATIONS

The concentrated SDN controller has a worldwide system see, which makes it simple to control and deal with the system. AI strategies give the intensity of knowledge to the SDN controller by performing information investigation, organize streamlining, and computerized arrangement of system administrations. On the other hand, the learning capacity gives the SDN controller to self-sufficiently figure out how to make phenomenal. Choices to adjust to organize conditions. In this segment, we audit existing AI endeavors to address issues of routing in SDN.

A suitable routing system provides QoS / QoE prediction, asset board, and security. Here is a per-user overview of how the ML algorithm is applied in the area of SDN. When dealing with advanced routing calculations, several goals were considered: Liu et al. [28] The focus is on minimizing network congestion and load imbalances in SDN networks. These basically represent calculations that detect and track elephant currents (large amounts of current). The stream is only recognized as an elephant if the number of bytes in the TCP cradle exceeds the predefined limit. All elephant streams are divided into different sub-streams.

To improve load parity and connection usage, each sub- stream is directed through various ways relying upon interface use. Experimental outcomes show that the recommended directing calculation beats single way steering and equivalent expense multi-way steering (ECMP) [30] as far as system throughput and connection use.

In any case, the multifaceted nature of the proposed algorithms is more prominent than O(n3). Subsequently, it takes a generally long effort to identify elephant streams and figure sending rules which can lessen foundation's presentation. Curtis et al. [29] center around limiting the remaining burden of the system controller and present Mahout, a traffic the board framework that can distinguish the class of the streams and course them in like manner. Mahout perceives. Elephant streams toward the end have through a shim in the working framework. On the off chance that measure informationi in TCP cradles surpasses an edge, at that point this type of flow is known as elephant flow and when it is distinguished, the system controller is seeni by utilizingi the in-band flaggingi instrument. The controller essentially registers the sending rules dependent on connect use, the connection with the most reduced use is chosen first. Mice streams are steered utilizing ECMP. Test results show that Mahout outflanks. Hedera [31] as far as the time controller remaining burden.

One of the biggest drawbacks of machine learning algorithms is the inability to easily deploy these techniques in real time. Machine learning provides high accuracy in analyzing network traffic. It is widely used in intrusion detection systems, as shown in [32]. However, this is computationally laborious and time consuming. As shown in [33], for example [34] attempted a calculated fallback, but there is no component that successfully integrates the new state, and execution time is important. When the system size increases exponentially. Unlike traditional machine learning approaches, reinforcement learning has variations. Reinforcement learning has long been used in system administration, [34] have applied intelligent learning modules inside switches, to attempt to join dynamic boundaries like blockage into choosing the progression of packets, and in any case, they could have constrained achievement on account of the exploratory runs included are excessively exorbitant, and the calculation utilizes an insatiable methodology, coming about in the [35]. Not with standing, tests directed by [36] show how the computationally serious nature of these calculations ends up being a barrier incapable of organization of such methods in the space of systems administration.

4. CONCLUSION

This article gave a situation of current ML strategies applied to Software-defined networking. We started our conversation with SDN which is another and well-known worldview of systems administration. It also focuses on SDN incorporation information. Since then, machine learning algorithm reviews have been introduced. At this point, we have elaborated on how ML calculations are applied to SDNs in terms of traffic order, line advances, QoS / QoE expectations, executive benefits, and security.It also covers the vast research agenda and future research topics in ML-based SDN, including best-in-class dataset preparation, distributed multi-controller phase, improved system security, cross-layer network expansion, and gradual adoption of SDN. Also talked.
In short, research on the application of ML algorithms in SDN is extensive and various problems still remain. Overall, it makes sense for the framework to meet these challenges and move forward. In this article, we will quickly investigate how ML algorithms work and when to use them to solve SDN problems.

ACKNOWLEDGEMENTS

We would like to acknowledge the funding support from Jaipur National University, India. My special thanks are extended to all professor from Computer Science Department, School of Engineering and Technology, Jaipur National University, India.

REFERENCES

[1] "Open Networking Foundation," Jun. 2014. [Online]. Available: https://www.opennetworking.org/

[2] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," IEEE Commun. Magazine, vol. 51, no. 7, pp. 36–43, July. 2013.

[3] "Open vSwitch," May. 2018. [Online]. Available: https://www.openvswitch.org/

[4] "Indigo: Open Source OpenFlow Switches," May. 2018. [Online]. Available: http://www.projectfloodlight.org/indigo/

[5] "Pantou: OpenFlow 1.3 for OpenWRT," May. 2018. [Online]. Available: https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow- 1.3-for- OpenWRT

[6] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA: An open platform for gigabit-rate network switching and routing," in Proc. IEEE MSE'07, San Diego, CA, USA, June 2007, pp. 160–161.

[7] M. B. Anwer, M. Motiwala, M. b. Tariq, and N. Feamster, "Switch- Blade: A platform for rapid deployment of network protocols on programmable hardware," in Proc. ACM SIGCOMM'10, New Delhi, India, 2010, pp. 183–194.

[8] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang, "ServerSwitch: A programmable and high performance platform for data center networks," in NSDI, vol. 11, 2011, pp. 2–2.

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, July. 2008.

[10] M. McCauley, "About Pox," 2013. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[11] Floodlight, "Project Floodlight open source software for building software defined networks," 2012. [Online]. Available: http://www.projectfloodlight.org/

[12] Ryu, "Ryu SDN Framework," 2013. [Online]. Available: http://osrg.github.io/ryu/

[13] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in Proc. IEEE WoWMoM'14, Sydney, NSW, June. 2014, pp. 1–6.

[14] D. Erickson, "The Beacon OpenFlow controller," in Proc. ACM SIGCOMM Workshop HotSDN'13, Hong Kong, China, 2013, pp. 13–18.

[15] E. Alpaydin, Introduction to Machine Learning. MIT Press, 2014.

[16] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," Emerging Artificial Intelligence Applications in Computer Engineering, vol. 160, pp. 3–24, 2007.

[17] J. Friedman, T. Hastie, and R. Tibshirani, The Elements of Statistical Learning. Springer Series in Statistics New York, 2001, vol 1.

[18] X. Zhu, "Semi-supervised learning literature survey," Citeseer, pp. 1– 59, 2005.

[19] X. Zhou and M. Belkin, "Semi-supervised learning," in Academic Press Library in Signal Processing. Elsevier, 2014, vol. 1, pp. 1239– 1269.

[20] D.-H. Lee, "Pseudo-label: The simple and efficient semi supervised learning method for deep neural networks," in Workshop on Challenges in Representation Learning, ICML, vol. 3, 2013, p. 2.

[21] H. Wu and S. Prasad, "Semi-supervised deep learning using Pseudo labels for hyper spectral image classification," IEEE Trans. Image Processing, vol. 27, no. 3, pp. 1259–1270, March 2018.

[22] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," IEEE Trans. Neural Networks, vol. 20, no. 3, pp. 542–542, 2009.

[23] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. MIT Press Cambridge, 1998, vol. 1, no. 1.

[24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237–285, 1996.

[25] C. J. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, no. 3-4, pp. 279–292, 1992.

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[27] Y. He, C. Liang, R. Yu, and Z. Han, "Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach," IEEE Trans. Network Science and Engineering, pp. 1–1, 2018.

[28] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo, and W. Dai, "Sdn based load balancing mechanism for elephant flow in data center networks," in 2014 International Symposium on Wireless Personal Multimedia Communications (WPMC), pp. 486–490, IEEE, 2014.

[29] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using endhost- based elephant detection.," in Infocom, vol. 11, pp. 1629–1637, 2011.

[30] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[31] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, et al., "Hedera: dynamic flow scheduling for data center networks.," in Nsdi, vol. 10, 2010.

[32] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in 2010 IEEE symposium on security and privacy. IEEE, 2010, pp. 305–316.

[33] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," Future Generation Computer Systems, vol. 92, pp. 59–67, 2019.

[34] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in Advances in neural information processing systems, 1994, pp. 671– 678.

[35] L. Li, Y. Zhang, W. Chen, S. K. Bose, M. Zukerman, and G. Shen, "Na¨ıve bayes classifier-assisted least loaded routing for circuit- switched networks," IEEE Access, vol. 7, pp. 11 854–11 867, 2019.

[36] S. Troia, A. Rodriguez, I. Mart´ın, J. A. Hern´andez, O. G. De Dios, R. Alvizu, F. Musumeci, and G. Maier, "Machine-learning-assisted routing in sdn-based optical networks," in 2018 European Conference on Optical Communication (ECOC). IEEE, 2018, pp. 1– 3.

[37] G. Stampa, M. Arias, D. Sanchez-Charles, V. Munt´es- Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software- defined networking routing optimization," arXiv preprint arXiv:1709.07080, 2017.

Authors

Mr. Vikas Verma is Asst. professor at Jaipur National University Jaipur India. He completed is Post graduation in 2012 his research interest is in Machine learning, Software Defined Networking.

Mr. Hement Mathur is Assot. professor at Jaipur National University Jaipur India. He completed is Post graduation in 2007 his research interest is in Machine learning, Computer Networking and Computer vision.