Enhancing dynamism in management and network slice establishment through deep learning

Rodrigo Moreira*†, Larissa Ferreira Rodrigues*, Pedro Frosi Rosa†, Rui L. Aguiar‡, Flávio de Oliveira Silva†
*Institute of Exact and Technological Sciences, Federal University of Viçosa (UFV), Rio Paranaíba/MG, Brazil
Email:{rodrigo, larissa.f.rodrigues}@ufv.br
†Faculty of Computing (FACOM), Federal University of Uberlândia (UFU), Uberlândia/MG, Brazil
Email:{rodrigo.moreira, flavio, pfrosi}@ufu.br
‡Telecommunications Institute (IT), University of Aveiro, Aveiro, Portugal
Email: ruilaa@ua.pt

2021 International Conference on Information Networking (ICOIN) | 978-1-7281-9101-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICOIN50884.2021.9333872

***Abstract*—With the variety of applications and the different user requirements, it is necessary to offer tailored resources efficiently not only in access but also in the core of the network. Inspired by the definition and standardization of mobile networks, especially 5G that focused on business verticals, the term network slicing has received numerous state-of-the-art efforts to materialize an approach that meets dynamism, programmability, and flexibility requirements. Leveraged by SDN and NFV technologies, network slicing is inspiring by resource sharing similar to virtual machine management, allowing standard network hardware to accommodate a wide variety of logical networks with specific requirements and data and control planes. However, state-of-the-art approaches do not address resource slicing at the core of the network in detail and appropriately. Therefore, we built NASOR to provide network slicing over the Internet data plane spanning across multiple domains through a segment routing and a distributed-based approach. Our approach excels those found in state-of-the-art by delivering an open policy interface that allows third-party applications to manage network slices dynamically. In this sense, this paper exploits this interface through a mechanism of convolutional neural networks that classifies network traffic, instructing the path-setting agent to be aware of application which predominantly runs on the network improving dynamism in the network slices deployment. Experiments showcase the convolutional neural network applicability and suitability as an enabling technology to enhance and instruct NASOR to establish network slices over multiple domains.**

***Index Terms*—SDN, NFV, Segment Routing, Network Sllicing, Deep Learning, Convolutional Neural Networks**

I. Introduction

To deal with different granularity levels of requirements that applications impose on the network, management, and orchestration mechanisms are increasing in the state-of-the-art [1], [2], [3]. The service-aware feature is massively present in the specification and standardization of mobile networks, especially at $5G$, which emerged network slicing as a consensual enabling technology [4], [5].

Network slicing inherits virtualization concepts in computing that provides resource sharing of standard hardware for tenants owners privately [6]. To employ sharing notions within networks, the community is proposing and combining enabling technologies in this ecosystem. Software-defined Networking (SDN) is a paradigm that decouples data and control plane of the network [7], providing an interface for programmability and customization of the desired behaviors for the network [8]. This technology has provided findings in numerous network management and operation approaches.

Jointly, Network Function Virtualization (NFV) has modified a way of delivering network functions, commonly provided through traditional vendors that offer specific network functions, separating software and hardware functions [9], [10]. Similarly, the segment routing which materializes its envisioning meets in active networks and source routing concept raising as an enabling technology increasing the dynamicity in packet forwarding in with a stateless control plane approach [11], [12]. Thus, combining SDN, NFV, and segment routing, many challenges were addressed, filling the management and orchestration gap in the network slicing ecosystem [13] materialized in the NASOR proposal [14].

Thereby, this paper presents the exploitation of the interface Open Policy Interface (OPI) of NASOR [14], which enables third-parties applications to change the network slicing path. We ran experiments to measure how potential third-party applications provide applicability and performance to instruct the network slice configure agent, taking into account the predominant traffic type running on interfaces where slices will be configuring. Hence, we combine PacketVision [15], previously proposed, with our newest OPI approach to materialize a smart network slice establishing.

The main contributions of this paper are summarized as follows: (1) Exploiting the applicability and suitability of the Open Policy Interface as a designed approach to enhance network slice deployment in NASOR ecosystem through third-party implementations; (2) The PacketVision method for building a dataset basing on the full-packet structure, considering both header and payload; (3) A quantitative comparison of the performance of three pre-trained CNN architectures in terms of classification accuracy, precision, recall, f1-score, and training time.

The remainder of this paper is organized as follows: Section II brings related work. Section III presents our approach to network slicing and traffic classification. Section IV explains our evaluation scenario. Section V discusses the results. Conclusion and future work are presented in Section VI.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.

II. Related Work

In [16], is proposing a traffic classification mechanism based on convolutional neural networks. The proposed mechanism extracts the payload bits from the packets and divides them into a specific ratio to build a square matrix. When bits are missing to establish a perfect division, bits-padding had been adding. Thus, the representation of an application class corresponds to the figures set, wherein each pixel of a specific figure is the result of a mathematical operation that combines all the payload bits of the packet into a single pixel. In this sense, to build a class figure set, it is necessary to take several flows of the same application into account. Our proposal goes further by considering the fully-packet structure.

A proposal that also uses convolutional neural networks to provide application classification and traffic categorization is available in [17]. The proposal integrates feature extraction and classification through deep neural networks of the type stacked autoencoder (SAE) and convolutional neural network (CNN). The method takes into account, in the pre-processing phase, the entry of the packet capture (*pcap*), which proceeds with header extraction, modification, and bits normalization, and IP address masking. After being processed and converted into bit strings, the data feeds into the neural network input. In contrast to the present proposal, the authors assume the bit chain that represents a semantics of the packet data as an input for learning the neural network.

The approach seen in [18] combines CNNs and Long Short Term Memory (LSTM) for traffic classification considering 108 classes. The classifier mechanism is flow-based, and its pre-processing approach considers the initial 20 packets of a flow. This approach relies on feedback that the final CNN tensor transforms the vector of characteristics into a matrix acting as an input to the LSTM. In contrast to the proposed work, the [18] approach considers as inputs to the machine learning mechanism a set of information extracted from flows.

Other detection approaches for security and privacy purposes had been seeing in [19]. The proposal brings a generalist mechanism for classifying traffic based on deep learning and a taxonomic model for schematizing state-of-the-art contributions. Additionally, the framework includes the pre-classification stage for cleaning and normalization. In contrast to the [19] proposal, we have advanced the state-of-the-art by materializing the implementation of an online traffic classification approach.

The proposed presented in [20] combines the Reproducing Kernel Hilbert Space (RKHS), and CNN approaches to provide IP traffic classification. The approach consists of extracting statistical flow characteristics in tuple format, considering the mathematical model RKHS transforming them into six channel images. However, this study raises questions about the overfitting possibility since the packets of the same class lead to having the same size and sequencing behavior. Our proposal goes beyond the [20] approach by providing a traffic classification mechanism in a free and straightforward flow pre-processing way.

III. Proposal

This section presents the integration of a third-party CNN-based approach with NASOR through OPI for instructing network slice establishment spanning across multiple domains taking into account the kind of application running on the network.

A. NASOR Overview

The state-of-the-art does not fully deliver network slicing at the core of the network. In this sense, the Network and Slice Orchestrator (NASOR) [14] approach makes it possible for the network to be shared and allocated to multi-tenants similar to what befalls in the management of virtual machines. The challenges facing network slicing are diverse, especially in a scenario of multiple domains that have technological and political autonomy.

The NASOR framework establishes logical connectivity that spans multiple domains, known as Autonomous Systems, and each logical connectivity has independent management and control available to owners. To act as a network slice manager, NASOR comprises SDN, NFV, Segment Routing, and distributed domain information repository technologies.

As depicted in Fig. 1, NASOR has entities settled in blocks that communicate and perform instructions to provide logical connectivity for a user. The first NASOR an independent entity within each domain, which implements network slicing based on Segment Routing. The entity is responsible for managing and handling requests for creating network slices, exchanging computer and network resource information for each domain, and instantiating NANO, which is the network slice manager made available to owners/users.

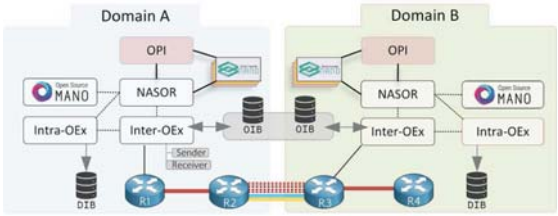

Fig. 1. NASOR Framework Overview.

B. The Open Policy Interface

The Open Policy Interface (OPI) is a set of methods that allows third-party implementations to be attached to NASOR to provide a specific function. Fig. 2 depicts the components build-block of the NASOR architecture and its third-party applications enablement. This relationship happens through the RestAPI interface exchanging messages such as topology path parameters. The abstraction that this interface proposes enables third-party implementations to operate on a data structure that defines the path to the network slice by exchanging asynchronous messages on the interface. In this context, dynamism in the management and implementation of a network slice is guaranteeing.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.

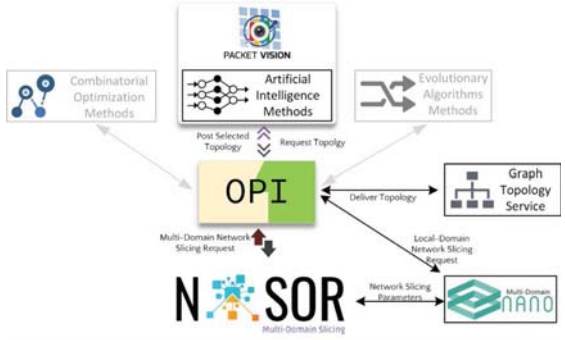

Fig. 2. NASOR Third-party applications enablement.

To increase dynamism in network slicing, especially in specifying alternative paths to routing algorithms, NASOR has two approaches for defining paths between multiple domains. First, one considers the data plane path provides through routing algorithms, such as BGP and OSP. In the second one, to exploit the OPI, we combine a third-party mechanism based on convolutional neural networks (PacketVision) to choose the network slice path, taking into account the predominant application running on the network.

C. Data Processing Method

Building the traffic classification mechanism and coupling its functionalities with the NASOR through OPI required data transformation in a preliminary phase. Our approach considered two datasets from different sources to construct the image classes according to their traffic characteristics. In the training and validation stage, the deep learning mechanism operates over a dataset intending to train the model to be used later in the classification.

The classes which comprise the dataset are the Bit Torrent and DNS classes extracted from the dataset available at [21]. Additionally, we built VoIP class from two entities communicating by ≈ 90 seconds by voice with data chunks processed by the g711 codec through a network slice deployed through NASOR. The IoT class comprises our dataset, including packets originated by 27 heterogeneous devices [22]. The final dataset generated from the raw data structure contains 5797 images, divided into Bit Torrent (1217), DNS (1412), VoIP (1320), and IoT (1848).

To connect the traffic classification into OPI, a CNN model had trained and validated. Subsequently, the data collection mechanism, built and accessed by the NASOR of each domain through gRPC and *pcaplibrary*, collects and forwards to the classifier, which returns the packet class. Therefore, the training and validation phase of the model befalls only once, making the classification later on-the-fly.

The packet capture step happens through the Wireshark tool, enabling the evaluation and export of the packet contents, especially in raw format. After collected, raw data are extracting from each packet, including headers and payloads in a Byte Array format. These bytes are organized in a decimal-matrix and following changed into a data model suitable for machine learning training.

Packets with the same source or destination may eventually have the related bytes in the same place, especially standardized fields such as TCP or UDP control headers, implying in pixels placing at the same location. Therefore, we proceed with the shuffling of information in the matrix of decimal numbers. This approach does not increase or provide data loss on the drawing-packet, but avoid bias, overfitting providing a classification on-the-fly.

The creating image dataset relies on transforming matrices, including decimal numbers into drawing-packet, and placing it correctly according to their traffic. Following the drawing-packet dataset and their classes have been built, it is possible to proceed with the training and validation mechanism of the traffic classifier, which takes into account the gray-scale drawing-packet dataset.

Fig. 3 depicts the construction phase for traffic class building. Extracting raw data allows collecting from packets hexadecimal data in the byte-array format. It is the leading information that enables our method to build and train a deep learning model comprising a PacketVision dataset. Therefore, the packet represents a hexadecimal byte-array containing its headers and payload.

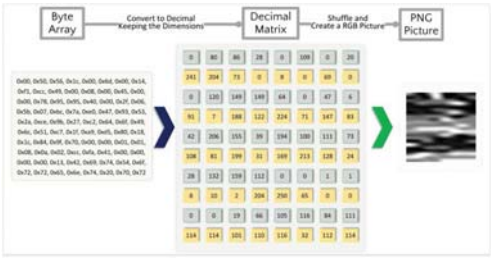

Fig. 3. Drawing-packet transformation method [15].

The PacketVision converts hexadecimal raw-data to decimal notation in the form of a square matrix. When the number of bytes precludes build an array, padding bytes are added to the end of the array until it is possible to construct it. In order to avoid bias, it was standardized to add *0xFF* of padding to the end of the array until a matrix is possible to build.

The format that the *pcaplibrary* exports byte-array packets straight implies the size of the matrices. Thus, the difference in size from one packet to the other is represented in the matrix with rows and never columns, so the notational size of the matrix is $n \times 8$. An 18-byte packet will be a 3×8 matrix with bytes padding in the last line. Another packet of 136-byte generates a 17×8 matrix without additional padding bytes. There is no bytes padding in the last line when the number of bytes in the array is precisely $n \times 8$.

PacketVision performs a normalization of the decimal matrix, allowing the numbers ranging from 0 - 255. In the next step, the PacketVision shuffles the decimal numbers, according to Poisson distribution, avoiding bias or overfitting in the model since bytes eventually stay at fixed locations implying

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.

in similar pixels locations. Further, the transformation of the shuffled-decimal matrix into a grayscale *PNG* figure. The end of the last step has the output of a set of images organized by the class representing a traffic class data suitable for input in the convolutional neural networks model.

Transforming the raw data into an image is necessary because the learning model requires data suitable for machine learning. Alternative approaches considering textual semantic representation or flow statistics are known [23]; however, our approach goes further, relying on the capabilities of the graphics processing hardware for traffic classification.

Fig. 4 depicts an example of drawing-packets produced through PacketVision. The figure brings a drawing-packet for each class considered in this paper; this set of drawing-packet is presenting to the deep learning mechanism in the training phase. It is possible to point out the toughness of the classification problem since the drawing-packet has varied features, even though they belong to the same class.

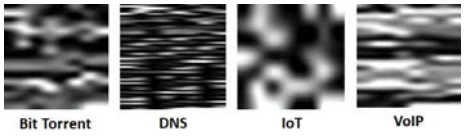

Fig. 4. Drawing-packet examples.

D. Convolutional Neural Networks

To exploit OPI through a third-party application, we propose a CNN network traffic classification. Deep Learning is a field of Artificial Intelligence, specifically Machine Learning, that uses multi-layer neural networks to learn features and classifiers in different layers, at running time, and does not require handcrafted feature extraction [24]. All deep learning-based networks, especially CNN, allow extracting high-level hierarchical representations of the data through multi-stage image processing [25].

We exploit OPI with three CNN architectures as third-party applications, and we chose them for experimental comparison based on their past performance in image classification tasks. The CNNs are: ResNet-50 [26], SqueezeNet [27], and VGG-16 [28]. Also, the training of all CNNs was performed using fine-tuning strategy [24] over models pre-trained with ImageNet.

IV. Evaluation

To compare the CNNs architectures to connects it to OPI, we trained and tested using a stratified k-fold cross-validation method [29] because this method is more robust to outliers and eventual overfitting. The 5797 images in the dataset were randomly sampled and partitioned into five folds. At each iteration of the cross-validation, one of the folds was selected for testing the trained model, and the $k-1$ folds were used for training. This procedure is repeated k times, alternating the testing folds. Thus, we ensure that each image will participate in the training process and will also be part of the test.

We measure the average of the four performance indices based on the number of true positives (TP), true negatives (TN), false positives (FP), and false-negative (FN) classifications obtained from the confusion matrix.

Our experimental evaluation aims to answer the following questions: (1) What is the most appropriate method based on convolutional neural networks to classify network traffic taking into account the metrics of accuracy, precision, recall, and f1-score and the dataset and the proposed scenario? (2) In terms of computational cost, what is the training time for each traffic classification approach? (3) Is OPI a suitable and scalable approach to increasing dynamism in management and network slice establishment between multiple domains?

V. Results

We carried some experiments to answer the previous questions, and their results are available in this section. All experiments ran using the Google Colaboratory cloud service with a machine Intel(R) Xeon(R) 2.20GHz processor, 12 GB RAM, and a GPU NVIDIA Tesla T4. The experiments were programmed using Python (version 3.6) and PyTorch 1.7 deep learning framework.

After partitioning the dataset into five folds using the stratified k-fold cross-validation method, we resized all images to 224×224 pixels based on bilinear interpolation; no information has been added or removed into images. The applied resize aimed only to adapt each image to serve as a suitable input into CNN architectures evaluated in this paper.

The data augmentation technique increased the training data artificially, without introducing labeling costs [30]. We performed the data augmentation using vertical and horizontal flips and rotating each of the original images around its center through randomly chosen angles of between $-10°$ and $10°$.

We trained the CNN architectures using Adam [31] optimizer, with a learning rate of 0.001, batch size of 32, and 30 epochs. Further, we measure the standard deviation of three CNNs architectures, and their results are as follows: VGG-16: 0.042; ResNet-50: 0.018 and SqueezeNet: 0.016 (for accuracy). These values indicate that our results, especially the training method, are reliable.

Regarding the classifications performance, the Table I presents metrics such as accuracy, precision, recall, and f1-score. Accuracy refers to the ratio between the number of correct classifications and the total of samples. On the other hand, precision estimates among all positive classifications (TP) how many are accurate. The recall is the ratio between correct classifications (TP) on TP and FP simultaneously. Finally, the f1-score defines the weighted average between the Precision and Recall metrics.

As reported in Table I, the SqueezeNet performed better than its peers, considering the dataset, the proposed scenario. This performance relies on the architectural robustness, whose main feature is the modules called "fire" that are stacked and divided into compression and expansion of convolutional filters, allowing large feature maps and, thus, allowing the accuracy of 96.80%. Also, the results of SqueezeNet, designed

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.

TABLE I
5-FOLD AVERAGE VALUES OF THE PERFORMANCE INDICES.

CNN	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
ResNet-50	95.00	95.50	94.75	95.00
SqueezeNet	**96.80**	**98.00**	**98.00**	**98.00**
VGG-16	91.75	92.60	92.00	92.00

to run on top of low-cost compute resources, suggesting that our method is suitable to be used as an efficient and scalable network traffic classifier mechanism on edge. Besides, showcased the suitability of using CNNs as a third-party application to support NASOR in network management and slicing.

Combining third-parties CNNs traffic classifications with the OPI enhances the management and establishment of inter-domain network slicing. NASOR can use the predominant traffic characteristic in the path establishing phase. Besides, segmenting network traffic leads network management predictable.

For each CNN, we assess the learning behavior analyzing the loss and accuracy values during the training phase and consider the average of all k-fold iterations (Fig. 5). Throughout the training, the behavior of the loss function generated low values for all CNNs. This behavior suggests that the training did not overfit the data, thus retaining the generalization property of the CNNs.

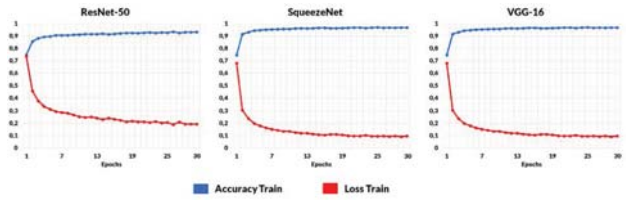

Fig. 5. Charts showing the evolution of accuracy and loss values for each CNN considering the average 5-fold training set.

Also, to explore the classification quality, confusion matrices for each CNN are presented in Table II. The confusion matrices describe several aspects of the classification problem investigated in this work showing the number of correctly classified and misclassified images.

Note that except for SqueezeNet, the CNNs misclassified a considerable proportion of the Bit Torrent images as IoT. Also, VGG-16 architecture misclassified $\approx 10\%$ of IoT images as DNS, which may require attention in future network traffic investigations. Other less relevant misclassifications occurred and varied with the CNN model.

We evaluated the training time for each CNN, taking into account the submission time until the end of the workload processing. Regarding traffic classification metric, the SqueezeNet model performed the best result in all the metrics evaluated. Besides, this model requires $\approx 2.7\times$ less computational time than VGG16 and $\approx 58.1\%$ less than ResNet-50. Hence, VGG-16 time consumption was 27:24 *min*, ResNet-50 17:24 *min*, and SqueezeNet 10:07 *min*.

If time and performance in the training phase are essential, SqueezeNet seemed to be the best choice. Additionally, it has proven that in training scenarios in distributed systems, SqueezeNet requires less communication between servers [32], less bandwidth to export a model in the cloud, and a suitable model for deployment on hardware with limited memory [27]. Also, the distributed nature of NASOR makes the classification mechanisms capable of operating in distributed scenarios compatible with SqueezeNet showcasing its suitability to work with OPI.

Accuracies upper 95% achieved from both ResNet-50 and SqueezeNet architectures showcased the applicability of fine-tuning on pre-trained models in the dataset. Using PacketVision as a third-party application, combined with data augmentation based on horizontal and vertical flips and random rotations, proved to be an effective method to provide traffic classification for applications on different channels.

The Future Internet architectures, including mobile networks, can also rely on efficient traffic classification mechanisms to provide better management and quality in the provision of tailored resources. In this sense, it is possible to infer from this experiment basing on the accuracy of the classification models, the applicability and suitability of OPI to propose dynamism in managing and establishing network slices between multiple domains.

VI. Concluding Remarks

This paper presents and evaluates the suitability of using a third-party application with NASOR to increase dynamism in managing and establishing network slices across multiple domains. The NASOR approach [14] had been paving the capability of handling network slicing challenges over the Internet data plane, enabling the NFV service chaining across multiple domains.

Exploiting the OPI, we overcome the traditional NASOR path-setting mechanism based on the data plane built by the routing algorithms. Hence, as explored in this article, OPI seems a suitable approach to add dynamism in establishing paths for network slices, mainly when basing the path on the predominant type of traffic classified through AI methods. We present a dataset construction method named Packet Vision, which relies on the full-packet structure.

Regarding traffic classification performance, the best result was achieved by SqueezeNet with an accuracy of 96.80%, enabling it to act as a third-party connected to OPI, instructing the NASOR in establishing the paths for the network slices.

For future work, we plan to evaluate the classification performance considering several network slices over the same channel and extending the set of classes to other traffic categories. We propose to assess and contrast a mechanism for defining paths based on combinatorial optimization and other machine learning techniques against those evaluated in this paper.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.

TABLE II
5-FOLD VALUES OF CONFUSION MATRIX FOR EACH CNN MODEL.

(a) ResNet-50

	Bit Torrent	DNS	IoT	VoIP
Bit Torrent	**1082**	6	123	6
DNS	1	**1353**	58	0
IoT	21	96	**1728**	3
VoIP	2	0	9	**1309**

(b) SqueezeNet

	Bit Torrent	DNS	IoT	VoIP
Bit Torrent	**1132**	2	82	1
DNS	1	**1393**	18	0
IoT	39	36	**1768**	2
VoIP	1	0	2	**1317**

(c) VGG-16

	Bit Torrent	DNS	IoT	VoIP
Bit Torrent	**1081**	27	106	3
DNS	27	**1351**	34	0
IoT	50	194	**1588**	16
VoIP	1	1	21	**1297**

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, "Nesmo: Network slicing management and orchestration framework," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 1202–1208.

[3] T. Taleb, B. Mada, M. Corici, A. Nakao, and H. Flinck, "Permit: Network slicing for personalized 5g mobile telecommunications," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 88–93, 2017.

[4] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, p. 1, 2016.

[5] J. Swetina, "Study on radio access network (ran) sharing enhancements, technical report (tr), 22.852, v15.1.0." [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=668

[6] Z. Shu and T. Taleb, "A novel qos framework for network slicing in 5g and beyond networks based on sdn and nfv," *IEEE Network*, pp. 1–8, 2020.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69–74, Mar. 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[8] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, p. 87–98, Apr. 2014. [Online]. Available: https://doi.org/10.1145/2602204.2602219

[9] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow world congress*, vol. 48. sn, 2012, p. 202.

[10] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[11] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," in *Proceedings DARPA Active Networks Conference and Exposition*, 2002, pp. 2–15.

[12] M. Xhonneux, F. Duchene, and O. Bonaventure, "Leveraging ebpf for programmable network functions with ipv6 segment routing," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 67–72. [Online]. Available: https://doi.org/10.1145/3281411.3281426

[13] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an sdn-enabled nfv architecture," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 187–193, 2015.

[14] R. Moreira, P. F. Rosa, R. L. A. Aguiar, and F. de Oliveira Silva, "Enabling multi-domain and end-to-end slice orchestration for virtualization everything functions (vxfs)," in *Advanced Information Networking and Applications*, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Cham: Springer International Publishing, 2020, pp. 830–844.

[15] R. Moreira, L. F. Rodrigues, P. F. Rosa, R. L. Aguiar, and F. d. O. Silva, "Packet vision: a convolutional neural network approach for network traffic classification," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2020, pp. 256–263.

[16] H.-K. Lim, J.-B. Kim, K. Kim, Y.-G. Hong, and Y.-H. Han, "Payload-based traffic classification using multi-layer lstm in software defined networks," *Applied Sciences*, vol. 9, no. 12, p. 2550, 2019.

[17] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, Feb 2020. [Online]. Available: https://doi.org/10.1007/s00500-019-04030-2

[18] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[19] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[20] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 1271–1276.

[21] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, "Is our ground-truth for traffic classification reliable?" in *Passive and Active Measurement*, M. Faloutsos and A. Kuzmanovic, Eds. Cham: Springer International Publishing, 2014, pp. 98–108.

[22] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2177–2184.

[23] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2019.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[25] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, "Everything you wanted to know about deep learning for computer vision but were afraid to ask," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, Oct 2017, pp. 17–41.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[27] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," 2016.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[29] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[32] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2592–2600.

Authorized licensed use limited to: b-on: UNIVERSIDADE DO MINHO. Downloaded on August 22,2025 at 19:41:47 UTC from IEEE Xplore. Restrictions apply.