Improving the network traffic classification using the Packet Vision approach

Rodrigo Moreira*†, Larissa Ferreira Rodrigues*
*Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa – UFV
Rio Paranaíba - MG - Brasil
Email: {rodrigo, larissa.f.rodrigues}@ufv.br

Pedro Frosi Rosa†, Flávio de Oliveira Silva†
†Faculdade de Computação - FACOM
Universidade Federal de Uberlândia – UFU
Uberlândia - MG - Brasil
Email: {rodrigo.moreira, pfrosi, flavio}@ufu.br

***Abstract*—The network traffic classification allows improving the management, and the network services offer taking into account the kind of application. The future network architectures, mainly mobile networks, foresee intelligent mechanisms in their architectural frameworks to deliver application-aware network requirements. The potential of convolutional neural networks capabilities, widely exploited in several contexts, can be used in network traffic classification. Thus, it is necessary to develop methods based on the content of packets transforming it into a suitable input for CNN technologies. Hence, we implemented and evaluated the Packet Vision, a method capable of building images from packets raw-data, considering both header and payload. Our approach excels those found in state-of-the-art by delivering security and privacy by transforming the raw-data packet into images. Therefore, we built a dataset with four traffic classes evaluating the performance of three CNNs architectures: AlexNet, ResNet-18, and SqueezeNet. Experiments showcase the Packet Vision combined with CNNs applicability and suitability as a promising approach to deliver outstanding performance in classifying network traffic.**

***Keywords*—Network traffic classification; convolutional neural networks; SDN; data augmentation; fine-tuning.**

I. Introduction

Classifying network traffic allows us to know the kind of application running on the network, benefiting the models for forecasting, capacity utilization, quality of service, security, and planning and management steps. Besides, in the frameworks of new communication, network architectures require intelligent entities to support resources management and operation. Traffic classification mechanisms are known and widely explored in the state-of-the-art, however, with the advent of convolutional neural networks (CNNs), new methods of training, validation, and classification are available, especially those based on images raising the opportunity to propose and evaluate mechanisms for network traffic classification [1] [2].

Among the known traffic classification mechanisms, we can categorize them as port-based, payload-based, machine-learning approaches based on statistics and deep learning [3]. In particular, CNNs demonstrate capabilities beyond its fields of action with highly accurate mechanisms for clustering and classifying medical images [4], biomolecular [5], environmental [6], and others contexts [7]. The success of CNNs is due to their ability to incorporate spatial context and weight sharing between pixels in order to extract high-level hierarchical representations of the data [8].

In this sense, we employ the capabilities of CNNs for processing packets of data communication networks. The graphics processing supported by GPU hardware surpasses the CPU-based processing because reducing the execution time [9]. Hence, the speedup of time-to-ready of traffic classification technologies is reducing [10], enabling faster classification.

Recent studies demonstrated effective results in network traffic classification using deep CNNs [1] [2] . However, these studies performed the classification splitting both header and payload of packets as a learning feature. In a real scenario, this approach may generate security and time issues, regarding the last one, it may increase the pre-processing time without guaranteeing gains in classification performance metrics.

In this paper, we proposed the Packet Vision: a method based on computer vision to generate images from both payload and packet header. Our main contribution relies on generating a single image representing all content of the network packet. Other approaches for traffic classification, such as those based on the packet signature [11], conflict with security, and privacy aspects, since information as the source and destination address, port, and transport protocol, to name a few are handling as plain text, making straightforward inference by malicious third-parties. Furthermore, a novel contribution of this paper is an evaluation of the performance of three state-of-the-art CNNs for the network traffic classification via training from scratch and fine-tuning.

Our results showcase the suitability and performance score of Packet Vision in generating and classifying images of packets from communication networks considering from raw-data. Besides, we considered three classification technologies based on CNNs, applying a hypothesis test to judge the performance between them.

The remaining of this paper is organized as follows: Section II surveys related work. Section III presents our approach for network traffic classification. The CNNs evaluated in this paper, and the protocol used in the experiments are presented in Section IV. Section V presents and discusses the results. Finally, we provide concluding remarks and future work agenda in Section VI.

II. Related Work

Lim et al. [2], proposed a traffic classification mechanism aims to improve the quality of service for applications without

interference from the network operator. Its structure generates a dataset containing images of flows analyzed over time intervals. The approach uses CNN and Long short-term memory (LSTM) to train and evaluate the classification performance using the F1-score metric. The proposed architecture considers three layers, the lowest containing data switches, including switches and hosts that exchange data between their own, on top of previous the control including classification mechanisms and traffic entities. The topmost layer allows the implementation of specific network behaviors based on the type of traffic.

The image generation mechanism for the dataset comprises capturing the flow: a set of packets with similar characteristics (source and destination host, port, and transport protocol) in a specific time interval. Therefore, for each packet of a flow, extracts its payload and performs mathematical operation over a set of bits to transform it into a single numerical value, consequently a single pixel. Thus, a single figure, containing many pixels, is the set of packet representing an application's flow. This approach does not carry out cross-validation and disregard the entire package structure, requiring additional computation in the processing step that consists of extracting the payload of each package.

Vasan et al. [12] proposed an architecture of CNN and evaluates the virtual threats as malware close classification real-time. The construction of the dataset transforms the binary signature of malware, which is an 8-bit vector into an 8-bit array, afterward in a grayscale figure and then applying a 2-D color map. The classification performance evaluation takes into account approaches with data augmentation and fine-tuning. Unlike the present proposal, this article proposes cross-validation to avoid bias and over model adjustment, besides there is no need to transform the image of the 2-D color maps dataset, maintaining performance.

Chen et al. [13] presents an IP traffic classification framework based on CNNs named Seq2Img. This approach consists of capturing the packets of a flow and extracting its characteristics and behaviors. A probability distribution model called Reproducing Kernel Hilbert Space (RKHS) is mandatory to construct the figures for each traffic class, consisting of the network protocols and popular social networking applications. Accuracy was the performance metric held in the validation of the traffic classification model. Unlike the present paper, the authors did not validate the proposal with hold-out, and the data collection mechanism depends on a third non-open source application. On the other hand, our approach consists of an open-source collector and does not handle images as flows and does not require processing with complex mathematical models.

Wang et al. [1] proposed a framework for classifying malicious traffic in domestic environments through home-gateway equipment containing an embedded traffic prediction mechanism. The mechanism based on CNNs is similar to ours because they take into account the figure from each package as a data suitable for Machine learning models. However, different from us in the pre-processing stage, the ethernet header of the package is removed. Besides, to avoid bias and overfitting in the training model, we applied, according to a probability distribution, we shuffle the image pixels of each class. Thus, packages containing the same source and destination address do not keep standardized pixels in predefined locations. The dataset images built from a set of packet captures of typical Internet standard applications.

Other works are known in the state-of-the-art, proposing a network traffic classification targeting security, quality of service enhancement, management, and others [14], [15], [16]. They vary in terms of the learning and validation method, also differs between strategies based on port, payload, statistics, CNNs, flows, and others [17]. The Packet Vision innovates by drawing the packets entirely, considering header and payload, and by creating a deep learning model considering those images generated through packets raw data.

III. Packet Vision

The resource sharing turn up in different ways in the literature. The architecture of the operating systems, especially those for time-sharing processing, has been inspiring new formats of resource sharing, impacting computing resources, and network sharing. Sharing network resource relies on to assign part of general-purpose hardware to a specific user while safeguarding essential aspects of isolation and guarantees. In the context of mobile networks, especially in the 5G standardization, sharing took the form of network slicing, which provides logical networks with independent data and control plans for users to meet specific application requirements.

Therefore, among the network slicing approaches rising the Network and Slice Orchestrator (NASOR) [18] that implements the network slicing beyond the mobile network ecosystem, providing logical connectivity over the Internet data plane. The NASOR ecosystem includes interfaces that facilitate network slice management, called the Open Policy Interface (OPI). The OPI interface allows third-party mechanisms to support network slicing and management. Consequently, we propose a component that performs this interface, offering traffic classification to lead the NASOR path configuring agent, called Packet Vision.

The Packet Vision is a method, originated from the drawing-packet action, capable of receiving a network packet in the raw format and transforming it into images considering both the header and the payload. After generating images, it is possible to classify them according to the traffic class. Traffic classes range across the network according to the overlying application. This classification guides the network slicing agent as to the path that logical connectivity must take along Internet routers. We present Packet Vision as a method of building a dataset of network traffic class images to train and evaluate deep learning algorithms. Hence, Fig. 1 depicts Packet Vision as a method for creating Datasets.

The first step comprises collecting network packets carried over a network interface. The open-source application Wireshark and its extension libraries allow collecting packet from a network interface without affecting the application. The Packet Vision handles packets traces from four sources, collected through the open-source tool Wireshark, containing *pcap* files for each traffic class.

The first class of traffic is the standard of IoT Applications containing around 27 heterogeneous devices such as sensors and actuators [19]. The second packet trace comprises conventional Internet applications containing DNS and BitTorrent classes [20], also available in *pcap* format.

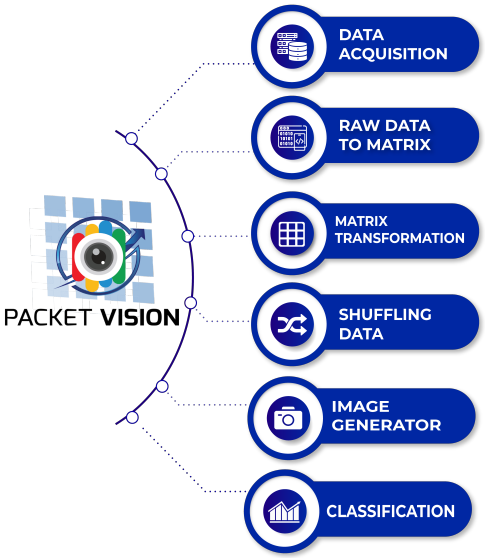

Fig. 1. Packet Vision proposed method.

The raw information of the packet available in this dataset had been processing in order to generate figures for each class. Finally, the third packet trace refers to network slice deployed through NASOR, considering three network domains [18]. Hence, a VoIP application providing communication between entities being in domain A targeting domain B communicating with voice chunks processed by codec G.711. Our method combines three packet traces making it possible to build a dataset of figures containing four traffic classes: BitTorrent, DNS, VoIP, and IoT.

The watcher captures the packets and presents it differently; bits is the conventional form of the physical layer. However, they are grouping in formats with semantic values, such as byte array, plain text, to name a few. Hence, the second step of the method consists of handling the data in raw format, distributed in an array of bytes, and transforming them into a matrix. In this sense, our method considers the data grouping model in the Array format, which presents the packet information in hexadecimal composition.

Regarding the second step, turn the hexadecimal byte array into a matrix whose size is $n \times 8$, where n represents the number of rows and 8 the number of columns according to Fig. 2. The matrix columns are 8 in size due to the native implementation of the Wireshark raw extractor library. The size of the packets, measured in bytes, varies among applications, so the method considers the number of columns fixed at 8, and the number of rows in the matrix is variable to accommodate the size of the packet in bytes. There are scenarios where the packet size in bytes is not $n \times 8$, requiring that bytes-padding appending at the end of the packet. We agree that bytes-padding is always *0xFF* for all traffic classes. Thus, when processing the matrix $n \times 8$ of hexadecimal and constructing the dataset organized in classes, these will contain figures of size $n \times 8$ pixels.

The third stage of the method considers as essential to convert the hexadecimal matrix, previously created, into decimal format. At the end of this step, the fourth shuffles the

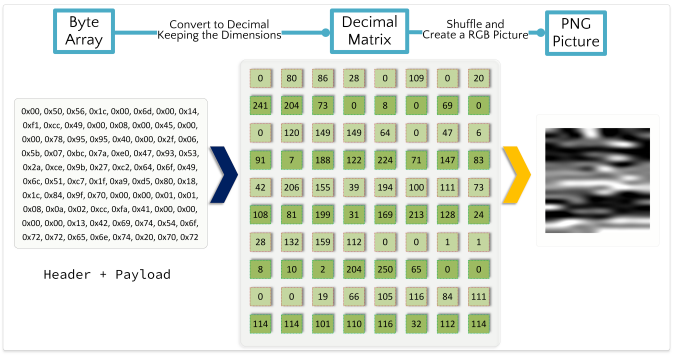

Fig. 2. Building Dataset.

decimal values of the matrix to avoid bias and overfitting in the deep learning model. Shuffling is mandatory to change the fixed place of packet headers, such as source host, destination host, port, to name a few. Our shuffling method held Poisson probability distribution over the decimal matrix, handling the security and privacy lack in the state-of-the-art. The decimal values representing the header may remain at fixed locations in the matrix, regardless of the package content. Therefore, the fourth stage performs a shuffling of the values according to a Poison probability distribution.

The fifth step of the proposed method consists of adding RGB channels according to each decimal in the matrix, maintaining the color intensity for the three channels. The fifth step brings PNGs figures representing the contents of the packet, including the headers and the payload as an image texture. Headers are the addressing information essential to the entire packet deliver, and the payload is the information carried.

The information about the created dataset has been summarizing in both Table I and Fig. 3, where the last one depicts examples of how Packet Vision can draw packets categorizing them into classes. This dataset are available at <https://romoreira.github.io/packetvision/> under open-source license.

TABLE I. DISTRIBUTION OF IMAGES BY CLASSES.

Class	Samples
Bit Torrent	1217
DNS	1412
VoIP	1320
IoT	1848
Total	5797

The sixth step includes training and validating the deep learning mechanism that uses the properly labeled figures from the created dataset. Many convolutional neural network architectures are known, so it is necessary to evaluate the performance of some to identify the most suitable for this kind of problem. After training and validating the learning model based on the figures generated through the raw packets, the characteristic of the current traffic on the network may be collected from a given network channel, by sample or for a determined time.

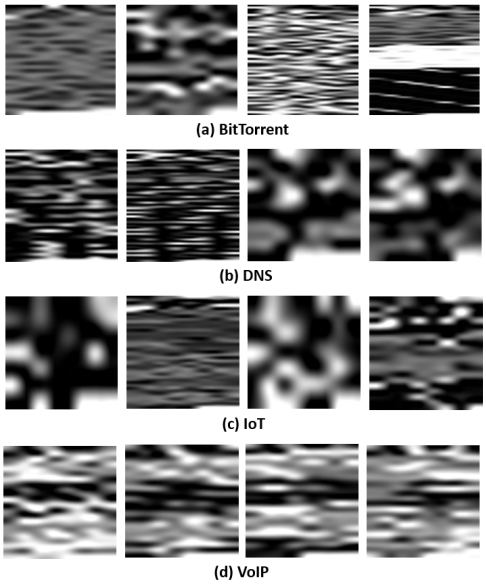

Fig. 3. Network packets samples generated from Packet Vision.

Other methods for building images from the packet are known [13], [21], [22], although they do not handle the complete packet structure. Alternatively, our method does not require the header and the packet payload separating in advance, causing additional processing. Besides, by shuffling the packet bytes in the matrix highlights our method regarding privacy, it is not straightforward to achieve the original semantics of the packet, including a source, destination, transport protocol port, and others from generated image.

IV. Classification Method

In this study, the classification was performed using CNNs, which uses multi-layer neural networks to learn features and classifiers in different layers, at running time, and does not require handcrafted feature extraction [23]. Three state-of-the-art CNN architectures were selected based on their past performance in image classification tasks: AlexNet [24], ResNet-18 [25], and SqueezeNet [26].

AlexNet [24] was the champion of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 and is responsible for the recent popularity of neural networks. This CNN has five convolutional layers, three max-pooling layers, two fully connected layers with a final softmax. It was a breakthrough architecture since it was the first to employ non-saturating neurons and dropout connections to prevent overfitting.

ResNet, presented in [25], was the champion of ILSVRC 2015 [27] and has several variations with 18 to 152 layers. This network has a series of residual blocks, each composed of several stacked convolutional layers. This configuration allows accelerating the convergence of the deep layers without overfitting. In this study, we choose to work with the ResNet-18 for the sake of simplicity.

SqueezeNet [26] has a compact architecture with approximately 50 times fewer parameters than AlexNet. This CNN reduces parameters through 1×1 convolutions and eight fire modules, which performs the functions of fully connected and dense layers.

We consider two strategies of training: from scratch and fine-tuning. In training from scratch, we initialize all parameters randomly, and during the training, the values of the parameters were learned directly from the dataset in all layers [23], achieving better results when compared with training based on fine-tuning since the CNN learns specific features [6] [4]. The fine-tuning strategy was performed over models pre-trained on the ImageNet dataset and consists of fine-tuning the parameters in the deeper layers [8]. For both training strategies, the dimension of the last fully connected layer was four, according to the number of classes.

In order to compare the CNN architectures, we trained and tested using stratified k-fold cross-validation method [28]. The cross-validation was repeated five times, for each iteration, one of the training folds is chosen for the test and the others for training. Also, was taken the average accuracy, precision, recall, and f1-score, measured from the confusion matrix [29].

V. Results and Discussion

All experiments were performed on a machine with an Intel i5 3.00 GHz processor, 16 GB RAM, and a GPU NVIDIA GeForce GTX Titan Xp with 12 GB memory. The experiments were programmed using Python (version 3.6) and PyTorch [30] (version 1.4) deep learning framework.

We trained the CNN architectures using Stochastic Gradient Descent (SGD) [31] optimizer, with a learning rate of 0.001, the momentum of 0.9, batch size of 32, and 50 epochs for both, training from scratch and fine-tuning. All images were resized to 224×224 pixels to adapt for the input of the CNNs evaluated. The training images had augmented through vertical and horizontal flips, rotating images around its center through randomly chosen angles of between 0° and 360°.

Our experiments aim to answer the following questions:

1) What is the highest classification performance among three evaluated CNNs?
2) Considering accuracy, training from scratch, and fine-tuning, what is the most suitable training method for this dataset?
3) Is the performance of pre-trained CNNs statistically equivalent?

To assess the impact of the training from scratch and fine-tuning, we analyze the classification performance of each CNN architecture according to metrics of accuracy, precision, recall, and f1-score. Regarding the classification performance, the Tables II and III presents the average 5-fold cross-validation for each CNN considering training from scratch and fine-tuning, respectively. As shown, the best performance results are achieving with the training from scratch. Consequently, the best result among the three has been obtaining by the AlexNet architecture, especially the strategy which use from scratch training.

Although the fine-tuning technique did not improve the performance indices compared to training from scratch, this approach requires less time to train the unfrozen layers and could be suitable in real scenarios (see Table IV). Thus, we

TABLE II. 5-FOLD AVERAGE VALUES OF THE PERFORMANCE INDICES FOR EACH CNN ARCHITECTURE TRAINING FROM SCRATCH.

CNN	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
AlexNet	**100.00**	**100.00**	**100.00**	**100.00**
ResNet-18	99.80	100.00	100.00	100.00
SqueezeNet	99.60	99.80	99.60	99.60

TABLE III. 5-FOLD AVERAGE VALUES OF THE PERFORMANCE INDICES FOR EACH CNN ARCHITECTURE TRAINING WITH FINE-TUNING.

CNN	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
AlexNet	95.40	96.00	95.80	95.80
ResNet-18	96.40	96.40	96.80	96.40
SqueezeNet	**97.60**	**97.80**	**97.40**	**97.60**

compared only the pre-trained CNNs in order to identify the best model.

TABLE IV. AVERAGE TRAINING TIME FOR EACH CNN ARCHITECTURE CONSIDERING BOTH TRAINING STRATEGIES.

	Training Time (minutes)	
CNN	**From-scratch**	**Fine-tuning**
AlexNet	16.21	06.21
ResNet-18	37.00	14.13
SqueezeNet	27.41	12.29

According to the results presented in Table IV, although training from scratch achieves high accuracy, the most suitable for this dataset considering the impact of computational cost is the SqueezeNet architecture trained with fine-tuning. Since, in real network traffic classification scenarios, approaches with lower computational cost are more appropriate.

To assess the performance, we carried Z-Test with 95% of confidence over samples of Table V, which contains accuracy obtained from each test set. Thus, considering AlexNet and ResNet-18, we raise the following hypotheses: H_0 – the performance of AlexNet is equal to or less than ResNet-18. On the other hand, H_a – the performance of AlexNet is higher than ResNet-18. Considering the sample space of size five, we can infer the observed $Z_{obs.}$ is lower than $Z_{crit.}$, leading us to accept H_0, implying that the performance of AlexNet is equal to or less than ResNet-18.

TABLE V. 5-FOLD TEST ACCURACY FOR EACH CNN ARCHITECTURE TRAINING WITH FINE-TUNING.

Fold	*AlexNet (%)*	*ResNet-18 (%)*	*SqueezeNet (%)*
1	93.00	95.00	96.00
2	97.00	97.00	98.00
3	98.00	98.00	98.00
4	93.00	95.00	97.00
5	96.00	97.00	99.00

Besides, we infer the performance of ResNet-18 and SqueezeNet, raising two hypotheses, namely H_0 – the performance of ResNet-18 is less than or equal to SqueezeNet. At the same time, H_a – the performance of ResNet-18 is higher than SqueezeNet. Considering a sample space with size five, and a normal distribution, the observed $Z_{obs.}$ is outside the critical region, which leads us to accept H_0, implying that ResNet-18 is less than or equal to SqueezeNet.

Hence, SqueezeNet architecture pre-trained with ImageNet had been performed better than or equal to its peers. These results suggest the suitability of Packet Vision to act as a traffic classifier mechanism and, eventually, enabling its embodiment on low-cost hardware such as Raspberry Pi.

Finally, considering the best result for each training strategy (from-scratch and fine-tuning), the charts in Fig. 4 show how each CNN architecture behaved during the training stage, considering the average loss and accuracy of the 5-folds. The results show that CNNs maintained the generalization property.

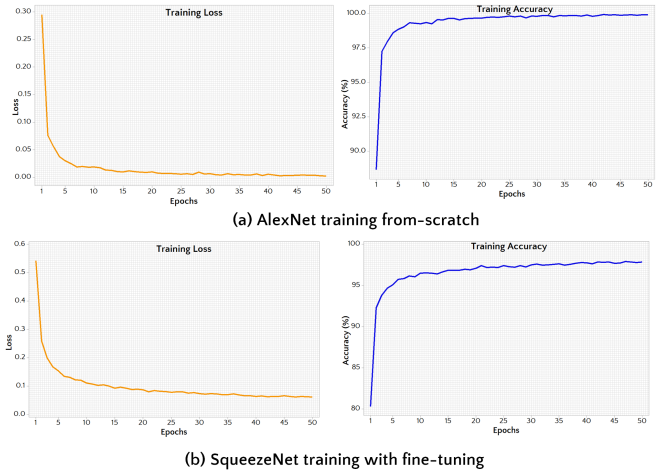

Fig. 4. Average 5-fold training loss and accuracy considering the best training strategy. (a) AlexNet training from-scratch; and (b) SqueezeNet training with fine-tuning.

VI. CONCLUDING REMARKS

This paper presents the Packet Vision method for building and evaluating datasets representing traffic on communication networks through CNNs. This method allows representing the raw-data of network packets in images for training and classification in a deep learning mechanism. The image creation mechanism considering the header and the payload advances the state-of-the-art since its peers consider only the payload, among other approaches such as the semantic and statistical representation of flows. Besides, our approach is suitable for classifying traffic with similar characteristics implying in challenging tasks, achieving excellent performances according to state-of-the-art metrics, and its implementation in the network being direct by handling the packets as they are.

Carried experiments showcase that SqueezeNet performance is at least equal or higher against AlexNet and ResNet-18 trained with fine-tuning, enabling us to answer questions about the quality of CNNs performance. Besides, we point out training approaches suitability for this problem, including a statistical test seeking possible performance equivalence. Also, unlike the approaches found in the state-of-the-art, the Packet Vision shuffling step enhances the privacy claim upon packets, avoiding fixed fields of the packets at the same pixel location, avoids rebuilding the original packet from the image.

We believe that Packet Vision is a robust application for the traffic network classification with a significant degree of innovation stemming from computer vision techniques applying to generate images from packets raw-data. Moreover, the Packet Vision seems suitable for future networks, such as 5G and beyond, whose take into account the security, privacy, and application-aware as a baseline.

As future work, we intend to exploit the Packet Vision approach to generate other traffic classes related to distinct applications, such as Remote Desktop Protocol (RDP), SSH, and social media. We are also planning to evaluate other CNN architectures, data augmentation strategies, and hyperparameter optimization.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN Xp GPU used for this research. And also, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] P. Wang, F. Ye, X. Chen, and Y. Qian. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, 6:55380–55391, 2018.

[2] Hyun-Kyo Lim, Ju-Bong Kim, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han. Payload-based traffic classification using multi-layer lstm in software defined networks. *Applied Sciences*, 9(12):2550, 2019.

[3] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*, 10(4):56–76, 2008.

[4] Larissa Ferreira Rodrigues, Murilo Coelho Naldi, and João Fernando Mari. Comparing convolutional neural networks and preprocessing techniques for hep-2 cell classification in immunofluorescence images. *Computers in Biology and Medicine*, 116:103542, 2020.

[5] Yukiko Nagao, Mika Sakamoto, Takumi Chinen, Yasushi Okada, and Daisuke Takao. Robust classification of cell cycle phase and biological feature extraction by image-based deep learning. *Molecular Biology of the Cell*, 31(13):1346–1354, 2020. PMID: 32320349.

[6] Keiller Nogueira, Otávio A.B. Penatti, and Jefersson A. [dos Santos]. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539 – 556, 2017.

[7] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27 – 48, 2016. Recent Developments on Deep Big Vision.

[8] 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)*, pages 17–41, Oct 2017.

[9] S. Potluri, A. Fasih, L. K. Vutukuru, F. A. Machot, and K. Kyamakya. Cnn based high performance computing for real time image processing on gpu. In *Proceedings of the Joint INDS'11 ISTET'11*, pages 1–7, 2011.

[10] S. Shi, Q. Wang, P. Xu, and X. Chu. Benchmarking state-of-the-art deep learning software tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104, 2016.

[11] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, MineNet '06, page 281–286, New York, NY, USA, 2006. Association for Computing Machinery.

[12] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138, 2020.

[13] 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)*, pages 1271–1276, 2017.

[14] S. Rezaei and X. Liu. Deep learning for encrypted traffic classification: An overview. *IEEE Communications Magazine*, 57(5):76–81, 2019.

[15] 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*, 5:18042–18050, 2017.

[16] F. Al-Obaidy, S. Momtahen, M. F. Hossain, and F. Mohammadi. Encrypted traffic classification based ml for identifying different social media applications. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–5, 2019.

[17] Murat Soysal and Ece Guran Schmidt. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Performance Evaluation*, 67(6):451 – 467, 2010.

[18] Rodrigo Moreira, Pedro Frosi Rosa, Rui Luis Andrade Aguiar, and Flávio de Oliveira Silva. Enabling multi-domain and end-to-end slice orchestration for virtualization everything functions (vxfs). In Leonard Barolli, Flora Amato, Francesco Moscato, Tomoya Enokido, and Makoto Takizawa, editors, *Advanced Information Networking and Applications*, pages 830–844, Cham, 2020. Springer International Publishing.

[19] 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)*, pages 2177–2184, 2017.

[20] Valentín Carela-Español, Tomasz Bujlow, and Pere Barlet-Ros. Is our ground-truth for traffic classification reliable? In Michalis Faloutsos and Aleksandar Kuzmanovic, editors, *Passive and Active Measurement*, pages 98–108, Cham, 2014. Springer International Publishing.

[21] T. Shapira and Y. Shavitt. Flowpic: Encrypted internet traffic classification is as easy as image recognition. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 680–687, 2019.

[22] L. Xu, X. Zhou, Y. Ren, and Y. Qin. A traffic classification method based on packet transport layer payload by ensemble learning. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2019.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] 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)*, pages 770–778, June 2016.

[26] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

[27] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[28] Pierre A. Devijver and Josef Kittler. *Pattern Recognition: A Statistical Approach.* Prentice-Hall, 1982.

[29] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York, NY, USA, 2000.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.