Packet Vision: a convolutional neural network approach for network traffic classification

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

***Abstract*—Network traffic classification can improve the management and network service 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 which can transform them 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 surpasses those found in the state-of-the-art, considering classification performance and regarding the fully-packet structure characteristic, delivering security and privacy by transforming the raw-data packet into images. Besides, we built a dataset with four traffic classes and evaluated three CNNs architectures, considering performance and the exploitation of training from scratch, fine-tuning and hyperparameter optimization. Experiments showcase applicability and suitability when combining Packet Vision with CNNs, which seemed to be a promising approach to deliver outstanding performance in the classification of network traffic.**

I. Introduction

Classifying network traffic allows us to know the kind of application running on the network, benefiting the forecasting models, capacity utilization, quality of service, security, and planning and management steps. Besides, in new communication frameworks, network architectures require intelligent entities to support resource 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 training methods, validation and classification are available, especially those based on images, which creates the opportunity to propose and evaluate mechanisms for network traffic classification [1] [2].

The known traffic classification mechanisms can be categorized 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 [4], biomolecular [5], environmental [6], and other images [7]. The success of CNNs results from their ability to incorporate spatial context and weight sharing between pixels 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 by reducing the execution time [9]. Hence, the speedup of time-to-ready of traffic classification technologies is reduced [10], which allows faster classification.

Recent studies demonstrated effective results in network traffic classification which the use of deep CNNs [1] [2] . However, these studies performed the classification by 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 latter, it may increase the pre-processing time without guaranteeing gains in classification performance metrics.

This paper presents the Packet Vision: a computer vision method to generate images from both payload and packet header. Our main contribution is the development of a single image representing all content of the network packet. Other traffic classification approaches, such as those based on the packet signature [11], struggle with security and privacy aspects. Sensitive information, such as source and destination address, port and transport protocol, to name a few, are handled 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 CNNs for the network traffic classification via training from scratch and fine-tuning. Besides, we found, quantified and presented the fine-tuning through Bayesian hyperparameter optimization.

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

The remaining of this paper is organized as follows: Section II survey 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 that aimed to improve the quality of service for applications, Remote Desktop Protocol (RDP), Skype, SSH, BitTorrent, HTTP-Facebook, HTTP-Google, HTTP-Wikipedia and HTTP-Yahoo. Their solution 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 comprises the following layers: network data switches, classification mechanisms and traffic entities and at the top, the implementation of specific network behaviors based on the type of traffic.

The dataset generation mechanism comprises capturing the network 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, they extracted the payload and performed a mathematical operation over a set of bits to transform it into a single numerical value. Thus, a single figure, containing many pixels, is the set of packet representing a network flow. In [2], they do not carry out cross-validation and disregard the entire package structure, which requires additional computation in the processing step that consists of extracting the payload of each package.

In [12] it is proposed and evaluated a CNN performance in virtual malware threat classification close to just-in-time. The dataset building transforms the binary signature of malware, comprising an 8-bit vector into an 8-bit array, afterward in a grayscale figure, and then a 2-D color map is applied. The classification performance evaluation takes into account approaches with data augmentation and fine-tuning. Unlike the present proposition, 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 map dataset, thus maintaining performance.

A proposal that also uses convolutional neural networks to provide application classification and traffic categorization is available in [13]. 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, bit normalization and IP address masking. After being processed and converted into bit strings, the data feed into the neural network input. In contrast to the present proposal, the authors assume the bit chain representing the semantics of the packet as an input for the neural network.

The paper [14] describes an IP traffic classification framework based on CNNs named Seq2Img. Their method captures the packets of a flow, extracting its characteristics and behaviors. A probability distribution model termed Reproducing Kernel Hilbert Space (RKHS) is mandatory to construct the figures for each traffic class, which consist 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. However, our approach consists of an open-source collector and does not handle images as flows and nor requires processing with complex mathematical models.

Similar to [12], the paper [1] describes 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, since it takes into account the figure from each package as a data suitable for Machine learning models. However, unlike our work, in the pre-processing stage, the ethernet header of the packet is removed. Besides, to avoid bias and overfitting in the training model, we shuffle the image pixels of each class. Thus, packages containing the same source and destination address do not keep patterned pixels in predefined locations. The dataset images are built from a set of packet captures from typical Internet applications, such as VoIP and BitTorrent.

Other works are known in the state-of-the-art, which propose a network traffic classification targeting security, quality of service enhancement and management, among others [15], [16], [17], [18]. They range in terms of the learning and validation method and also differ between strategies based on port, payload, statistics, CNNs and flows, among others [19]. The Packet Vision innovates by drawing the packets entirely, transforming network packets into figures, considering header and payload, and by creating a deep learning model, considering those images generated through packets raw-data.

III. The Packet Vision Approach

Network resource sharing is presented in different ways in the literature. The operating systems architecture, especially those for time-sharing processing, has influenced new resource sharing formats, computing resources and network sharing. Sharing network resources relies on assigning 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 found in the state-of-the-art, the Network and Slice Orchestrator (NASOR) [20] aimed to implement 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 Open Policy Interface (OPI). OPI 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. Fig. 1 depicts NASOR entities settled in blocks that communicate and perform instructions to provide logical connectivity for a user. The Packet Vision aimed to classify the traffic in the path established through NASOR.

The action of transforming network packets into images has been performing by the Packet Vision method. The method

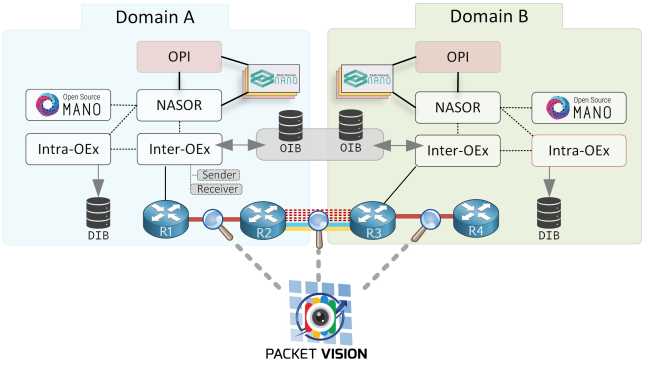

Fig. 1. Combining NASOR Framework and Packet Vision.

enables receiving a raw packet, drawing it into a picture considering its hexadecimal values, which carries network packet semantics. 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 to the path that logical connectivity must take along Internet routers. Besides, we present Packet Vision as a method to build dataset of network traffic class images, which enables us to train and evaluate through deep learning algorithms. Hence, Fig. 2 depicts Packet Vision general method for creating datasets.

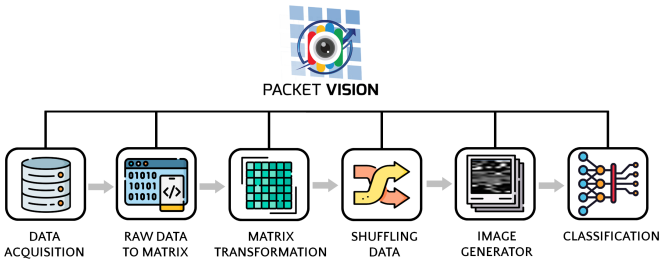

Fig. 2. Packet Vision proposed method.

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

We compose a dataset containing packet traces from different sources, implying in different classes. The first one relates to standard IoT Applications containing about 27 heterogeneous devices, such as sensors and actuators [21]. The second packet trace comprises conventional Internet applications containing DNS and BitTorrent classes [22], also available in *pcap* format.

The raw information carried on the packet, including header and payload, available in this dataset, was processed to generate figures for each class. The third packet trace refers to network slice deployed through NASOR, considering three network domains [20]. This trace refers to a VoIP application providing communication between entities placed in domain A targeting domain B, communicating with voice chunks processed by codec G.711. Packet Vision combines three different packet traces and builds a single dataset of figures containing four traffic classes: BitTorrent, DNS, VoIP and IoT.

According to Fig. 3, a watcher captures the packets and presents it differently; bits are the conventional form of the physical layer. However, they are grouped 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 raw format data, 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.

According to Fig. 2, the second step turns the hexadecimal byte array into a matrix whose size is $n \times 8$, where n refers to the number of rows and 8, the number of columns, as detailed in Fig. 3. The size of the packets, measured in bytes, varies among applications. Thus the Packet Vision considers the number of columns fixed at 8, while the number of rows in the matrix varies, so as to accommodate the size of the packet in bytes.

The reason the matrix column is eight refers to the format in which Wireshark library delivers the packet raw-data to Packet Vision. There are scenarios in which the packet size in bytes is not $n \times 8$, requiring to append bytes-padding at the end of the packet, namely, at the last row. 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.

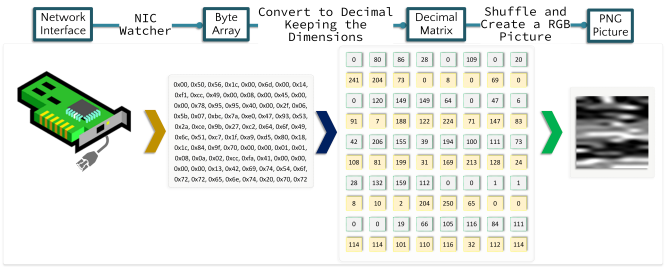

Fig. 3. Packet Vision: dataset builder.

The third stage of the method (Fig. 2) considers as essential to convert the hexadecimal matrix, previously created, into decimal format. At the end of this step, the fourth step shuffles the decimal values in the matrix (as depicted in Fig. 3) to avoid bias and overfitting on deep learning model. Shuffling is mandatory to change decimal information regarding packet header, such as source host, destination host and port, to name a few. Our shuffling method is based on the Poisson probability distribution over the decimal matrix, enduring the security and privacy lack observed in the state-of-the-art.

The fifth step (Fig. 2) of Packet Vision 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 packet contents, including both the headers and the payload, as a gray-scale picture. Headers are the addressing information essential to

the entire packet deliver, and the payload is the information carried.

We summarized the information about the created dataset in Table I, and we bring some examples of packet pictures that Packet Vision had drawn in Fig. 4. Besides, we detail the dataset structure, related publications, and made it available under an open-source[1] license.

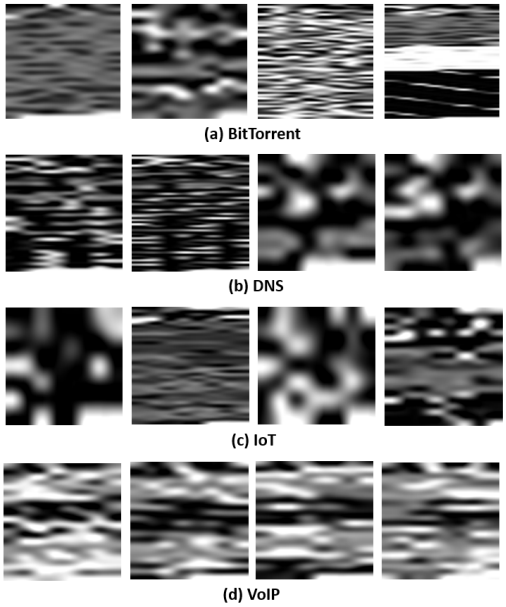

Fig. 4. Network packets samples generated from Packet Vision.

TABLE I. DISTRIBUTION OF IMAGES BY CLASSES.

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

The sixth step (Fig. 2) of Packet Vision entails training and validating the deep learning mechanism that uses the properly labeled figures from the created dataset. Several convolutional neural network architectures are known, so it is necessary to evaluate the performance of some of them 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 current traffic on the network may be collected from a given network channel, by sample, specific time and others.

As we can see at the state-of-the-art, other methods for building images from the packet are available [14], [23], [24], although they do not handle the complete packet structure. Alternatively, our method does not require the header and the packet payload separation in advance, which implies additional processing. Besides, shuffling the packet bytes in the matrix highlights the privacy of our method. It is not straightforward to achieve the original semantics of the packet, including a source, destination, transport protocol port, and others, from the generated image.

[1]https://romoreira.github.io/packetvision

IV. CLASSIFICATION METHOD

In this study, we carried out experiments to assess the classification performance using CNNs, which employ multi-layer neural networks to learn features and classifiers in different layers, at running time, and do not require handcrafted feature extraction [25]. Three CNN architectures were selected based on their past performance in image classification tasks: AlexNet [26], ResNet-18 [27] and SqueezeNet [28].

AlexNet [26] 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-18, presented in [27], was the champion of ILSVRC 2015 [29] 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 settled to work with the ResNet-18 for the sake of simplicity.

SqueezeNet [28] 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 perform the functions of fully connected and dense layers.

Fig. 5 depicts the main components of the CNN architectures evaluated in this paper: (a) convolutional layers of AlexNet; (b) residual block of ResNet-18; and (c) scheme of one fire module of SqueezeNet.

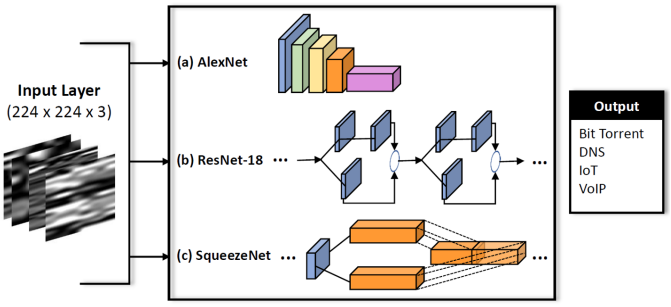

Fig. 5. A general representation of the building blocks of each CNN.

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 parameters were learned directly from the dataset in all layers [25]. The fine-tuning strategy ran over pre-trained models from the ImageNet dataset. All convolutional layers initialized with weights from the pre-trained model, and the fine-tuning was performed only in the deeper layers [8]. For both training strategies, namely,

from scratch and fine-tuning, the last fully connected layer had been set to four, according to the number of classes.

In order to compare the CNN architectures, we trained and tested using the stratified k-fold cross-validation method [30]. Cross-validation was repeated five times, for each iteration, and one of the training folds was selected for the test and the others for training. Also, the average accuracy, precision, recall, and F1-score were measured from the confusion matrix [31].

V. Results and Discussion

All experiments were conducted 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. Also, the experiments were programmed using Python (version 3.6) and PyTorch [32] (version 1.4) deep learning framework.

We trained the CNN architectures using Stochastic Gradient Descent (SGD) [33] optimizer, with a learning rate of 0.001, momentum of 0.9, batch size of 32, and 50 epochs. All images were resized to 224×224 pixels to adapt for the input of the CNNs evaluated. The training images had been augmented through vertical and horizontal flips, with rotating images around its center through randomly chosen angles of between 0° and 360°.

We propose some experiments aiming to answer the following questions:

1) What is the highest classification performance among the three evaluated CNNs?

2) Considering accuracy, training from scratch, fine-tuning and fine-tuning with hyperparameter optimization, what is the most suitable training method for this dataset?

3) Is the performance of pre-trained CNNs statistically equivalent?

4) What are the best values of hyperparameters (learning rate and momentum coefficient), which bring the highest classification performance?

5) How much does the hyperparameter optimization increase the performance of CNNs, considering fine-tuning approaches?

A. Training from scratch vs. fine-tuning

Aiming 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, Tables II and III present the average 5-fold cross-validation for each CNN, considering training from scratch and fine-tuning, respectively. As shown, the best performance results are achieved using the training from scratch. Consequently, the best result among the three was obtained by the AlexNet architecture, especially the strategy which uses 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 compared only the pre-trained CNNs in order to identify the best model.

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**

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

TABLE IV. Average training time for each CNN architecture considering training from-scratch and fine-tuning.

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

Aiming 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 that the observed $Z_{obs.}$ is lower than $Z_{crit.}$, which leads us to accept H_0, implying that the performance of AlexNet is less than or equal to 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

Additionally, we formulate two hypotheses to infer the performance of ResNet-18 and SqueezeNet, 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. Hence, we can infer that the ResNet-18 performance is less than or equal to SqueezeNet.

Therefore, SqueezeNet architecture, pre-trained with ImageNet, performed better than its peers. These results suggest the suitability of Packet Vision to act as a traffic classifier mechanism, and its eventual 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. 6 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.

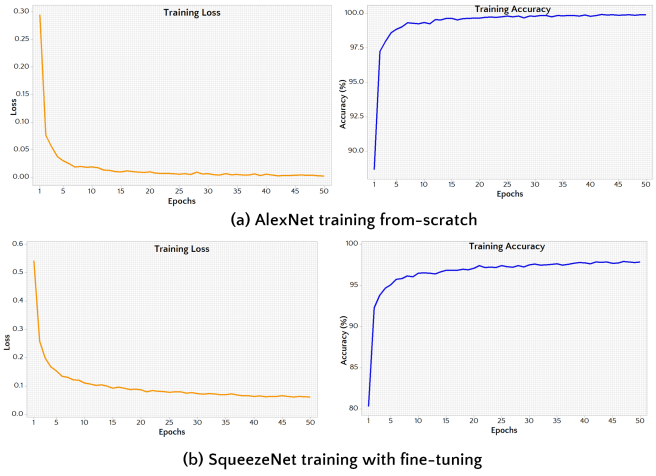

Fig. 6. Average 5-fold training loss and accuracy, considering the best training strategy. (a) AlexNet training from-scratch; and (b) SqueezeNet training with fine-tuning.

B. Fine-tuning with hyperparameter optimization

This experiment aimed to demonstrate the proper hyperparameter tunning to improve the performance of the CNNs. The selection of hyperparameter values was carried out as an optimization problem. Thereby, the validation error on the trained model is the objective function intended to minimize, and the hyperparameters are decision variables.

Properly tuning the learning rate improves learning efficiency, once it defines the adjustment levels to the weight connections and network topology applied in each training step. This is also fundamental to improve the runtime when using SGD [34]. Besides, the tuning of the momentum coefficient enables reducing noise and oscillations in the high-curvature regions of the loss function generated by SGD [8].

We applied the Bayesian optimization (using the Gaussian process as a probabilistic model [35]) to optimize the fine-tune values, namely the learning rate and momentum coefficient in pre-trained CNNs. The Gaussian function is a sequential model-based optimization (SMBO) used to find a globally optimal solution within the feasible region, based on previous observations. Besides, the Bayesian optimization is high efficiency, compared to the grid and random search approach [36] [37].

The hyperparameter optimization process for each fold took about 100 minutes for AlexNet, 220 minutes for ResNet-18, and 170 minutes for SqueezeNet. The Bayesian approach had also performed in two iterations, five steps of random exploration, and 50 epochs of training.

The hyperparameter space search is present in Table VI, and the searching values take into account an uniform distribution. Besides, Table VII presents the best values for each hyperparameter returned by the Bayesian algorithm. Those values are the best hyperparameter values that bring the highest classification performance.

TABLE VI. HYPERPARAMETER SEARCH SPACE USED FOR OPTIMIZATION.

Hyperparameter	**Value**
Learning Rate	$x \in [0.0001, 0.01]$
Momentum	$x \in [0, 1]$

TABLE VII. FINE-TUNING TRAINING-RELEVANT HYPERPARAMETERS OBTAINED FOR EACH CNN WITH BAYESIAN OPTIMIZATION.

		Hyperparameter	
CNN	**Fold**	**Learning Rate**	**Momentum**
AlexNet	1	0.0015433407950530326	0.9683901952093108
	2	0.002418215510161384	0.001523738687215559
	3	0.004228517846555483	0.7203244934421581
	4	0.00010113231069171439	0.30233257263183977
	5	0.004228517846555483	0.7203244934421581
ResNet-18	1	0.004228517846555483	0.7203244934421581
	2	0.004228517846555483	0.7203244934421581
	3	0.004228517846555483	0.7203244934421581
	4	0.00010113231069171439	0.30233257263183977
	5	0.004228517846555483	0.7203244934421581
SqueezeNet	1	0.00010000568478612993	0.000000574220821205
	2	0.00010113231069171439	0.30233257263183977
	3	0.002418215510161384	0.001523738687215559
	4	0.00022048625032683222	0.9857022918324738
	5	0.004228517846555483	0.7203244934421581

As shown in Table VIII, the Bayesian hyperparameter optimization improved the accuracy for all evaluated CNNs architectures. When we compare the results of fine-tuning training against and without hyperparameter optimization, it is possible to observe that the accuracy of AlexNet, ResNet-18 and SqueezeNet increased by 4.25%, 3.16%, and 1.45%, respectively. Also, the hyperparameter optimization in fine-tuning generates significant improvements, with results closer to the accuracies obtained through from-scratch training.

Our results suggest that the learning rate and momentum coefficient significantly impact the network training process.

Therefore, they have to be carefully tuned to achieve sactisfatory classification performance. Thus, the advantage of our approach is the selection of the best values for the hyperparameters, using Bayesian optimization. For our network traffic classification, with the hyperparameters set in Table VII, we had achieved the best average accuracy of 99.45% in the test set for AlexNet and ResNet-18 architectures. Besides, when we defined the learning rate and momentum coefficient with the values returned by the Bayesian algorithm, the spent training time was the same as the training based on fine-tuning.

TABLE VIII. 5-FOLD AVERAGE TEST ACCURACY FOR EACH CNN ARCHITECTURE AND ALL TRAINING STRATEGIES EVALUATED.

	Training Strategy		
CNN	**From-scratch**	**Fine-tuning**	**Fine-Tuning + Optimization**
AlexNet	100.00%	95.40%	99.45%
ResNet-18	99.80%	96.40%	99.45%
SqueezeNet	99.60%	97.60%	99.02%

VI. CONCLUDING REMARKS

This paper presents the Packet Vision method for building and evaluating datasets representing traffic on communication networks through CNNs. This method enables the representation of the raw-data of network packets in images for training and classification held by deep learning mechanisms. The image creation mechanism takes into account the header and the payload advancing 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 method seemed suitable for classifying traffic with similar characteristics implying in challenging tasks and achieving excellent performances, according to state-of-the-art metrics. Its implementation in the network is performed directly by handling the packets as they are.

Carried experiments showcase that SqueezeNet achieved higher or at least equal performance against AlexNet and ResNet-18 trained with fine-tuning, which allows us to answer raised questions about the quality of CNNs. Besides, we point out the suitability of training approaches for this problem, including a statistical test, seeking possible performance equivalence. Also, unlike the state-of-the-art approaches, the Packet Vision shuffling step enhances the privacy claim upon packets and avois fixed fields of the packets placed at the same pixel location. Besides, we found, quantified and presented the hyperparameters and measured the improvement they promoted in the performance of CNNs trained with fine-tuning.

We believe that Packet Vision is a robust application for the network traffic classification with a significant degree of innovation stemming from computer vision techniques applied to generate images from packet raw-data. Moreover, the Packet Vision seems suitable for future networks, such as 5G and beyond, which takes 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 different applications, such as Remote Desktop Protocol (RDP), SSH and social media. We are also planning to evaluate other CNN architectures, different data augmentation strategies and further optimization algorithms with more extensive hyperparameters set.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation through the donation of the TITAN Xp GPU used for this research. Besides, 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] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, Feb 2020.

[14] 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.

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

[16] 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.

[17] 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.

[18] Y. He and W. Li. Image-based encrypted traffic classification with convolution neural networks. In *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pages 271–278, 2020.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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.

[33] 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.

[34] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[35] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, page 525–532, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[36] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

[37] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.