When Deep Learning May Not Be The Right Tool For Traffic Classification

Kleidi Ismailaj*, Miguel Camelo†, and Steven Latré†
University of Antwerp - imec
IDLab - Department of Computer Science
Sint-Pietersvliet 7, 2000 Antwerp, Belgium
E-mail:firstname.lastname@{student.uantwerpen.be*, uantwerpen.be†}

***Abstract*—Traffic Classification System (TCS) allows inferring the application that is generating given network traffic. Other systems can use this information to enforce specific network policies on the analyzed traffic. In recent years, Traffic Classifier (TC) based on Deep Learning (DL) have outperformed traditional methods such as port-based and statistical Machine Learning (ML). Although these TC can achieve high accuracy on raw data, most of those works do not provide any reasoning or interpretation about how the trained model could achieve such performance. This lack of interpretability may lead to unpredicted behaviour of the systems that consume such information. To understand what the DL models are learning, we conduct a set of experiments reveal what the DL models are learning and we validate our reasoning by building and training simpler ML models that use the revealed features and could even outperform the DL models in some evaluations.**

***Index Terms*—Traffic Classification, Machine Learning, Deep Learning, Network Management**

I. Introduction

Data networks keep increasing their capacity to support the ever-increasing number of devices and applications. As the networks grow, managing and optimizing their resources become even more challenging [1]. A Traffic Classification System (TCS) is used to identify and classify among different types of traffic and provide a service to other systems to perform tasks such as Quality of Service (QoS) provisioning, billing, anomaly detection, resource usage planning, among others. Traditionally, behind a TCS there is a Traffic Classifier (TC) which task is given an input data, e.g., a set of features from a single packet or a flow of packets, discriminate among a set of traffic classes.

In the recent years, TC based on Deep Learning (DL) have outperformed the ones that are empowered by traditional methods such as port-based, Deep Packet Inspection (DPI), and flow-based traffic analysis using statistical Machine Learning (ML), and this result is more evident on traffic that is encrypted [2]. One of the reasons for such success is that DL models can automatically extract the features required to solve the given classification task with high accuracy from a large amount of raw data [3]. In other words, DL models offer the possibility of creating black-boxes of learning parameters, which can be trained to make the association between raw input data and target outputs on their own.

978-3-903176-32-4 © 2021 IFIP

However, even though many proposed DL-based TCs achieve high accuracy on raw packets, traditionally above 90%, most of those works do not provide any reasoning or interpretation about how the trained model was able to achieve such performance [4]. In this paper, interpretability of a model, a term that can be used interchangeably with explainability, is defined as *the degree to which an observer can understand the cause of a decision* [5]. As a consequence, if a ML model has higher interpretability, then the easier to humans to understand why some decisions or predictions have been made. We believe that this lack of interpretability of a ML-based TC may lead to several risks for the systems that consume such information, e.g., billing, when deployed in real environments as it would be hard to identify the root cause of poor performance in such systems. Moreover, as DL models are computationally expensive, using them as ”one size fits all” may leave them useless due to poor response time.

Motivate for the impact of using DL models for packet-based TC on raw input data, we conduct a set of experiments to measure the performance of the DL-based TCs when the input representation of the packet uses different combinations of the IP and transport protocol headers and payloads. As a dataset, we use the non-VPN part of the ISCX dataset [6], i.e., traffic with non-encrypted packets. These combinations allow us to understand the effects of given raw input representations on the DL model's performance. As a result, we can reveal the features that the DL model is learning. By combining data analysis techniques with experimental validations, we can build simpler ML models that use the revealed features and outperform the DL models in some evaluations.

The remainder of this paper is structured as follows. Related works are presented in Section II. The baseline DL-based TC system and the different modifications on the raw packet representation used for performance evaluations are presented in Section III. We show the DL model's performance evaluation results on both Traffic Characterization (coarse-grained) and Application Identification (fine-grained) tasks in Section IV. Conclusions and future work are presented in Section V.

II. Related Works

Lotfollahi et al. [2] present Deep Packet, a DL approach that can classify traffic with high accuracy. Their method can identify encrypted traffic and does not need any handcrafted features. The authors proposed two DL architectures, one based on Convolutional Neural Networks (CNNs) and one based on AutoEncoders (AEs). Both architectures are trained on the *ISCX VPN-nonVPN* dataset (captures on the data-link layer). After a pre-processing step, each IP packet are truncated/zero-padded to a fixed length of 1500 bytes, which is required by the selected DL architectures. The results show that Deep Packet can achieve an average score of more than 90% on two traffic classification tasks (traffic characterization and application identification. However, they do not provide conclusive evidence of what the DL models are learning to explain why their approach performs well.

Wang et al. [7] present a DL-based TC that uses one-dimensional CNN as learning architecture. The ISCX VPN-nonVPN dataset is used for validation. Four combinations of packet representation are considered: layer 7 of the ISO/OSI model (L7) or all protocol layers (ALL) (only the first 784 bytes), biflows, i.e., a 5-tuple composed by source IP, source port, destination IP, destination port, and transport-level protocol, and flows, i.e., the same 5-tuple but the source and destination IP / port are not interchangeable. Although the results show a very high accuracy, some experiments even with scores above 99%, the authors do not explain about what the model is being learned to justify the results. Moreover, the authors in [2] indicate that the packets in the ISCX VPN-nonVPN dataset contain unique source and destination IP addresses for each application. As a result, the proposed model may be simply learning this feature to classify the traffic.

Moreira et al. [8] introduce Packet Vision, a DL-based TCS capable of classifying traffic from packets raw-data. Their method considers both header and payload and consists of 6 steps: data acquisition, raw data to matrix transformation, matrix transformation, data shuffle, image generation, and classification. The first three steps are used to transform raw packets, usually in hexa or binary representation, to an $n \times 8$ decimal matrix. The fourth step provides change decimal information regarding packet header to enduring the security and privacy based on the Poisson probability distribution over the decimal matrix. Finally, the fifth step creates an RGB image of the shuffled matrix, which can be easily used for CNN-based classifiers (last step). Although the results show a very high accuracy, above 99%, on the datasets used for experimentation, the paper lacks interpretability of the model.

Aceto et al. [4] perform an experimental evaluation of different DL architectures applied to *mobile* encrypted TC. Mobile TC has extra challenges since there many apps to discriminate from, which are also subject to frequent updates (that may distort the previously configured models). The DL architectures are evaluated on three datasets collected by human users and labelled by the generating app and with *biflows* as traffic objects. Aligned with our work's motivation, one of the authors' conclusions is that when DL-based classifiers are fed with all the data contained in a packet, there is a high probability of misleading performance results and DL models can be reduced to statistical IP/port-based architectures. Moreover, the black-box nature of most of the DL-based classifiers makes difficult or not-at-all possible to predict the performance of the system, requiring to find the right balance between naive application and expertise-driven effort. In this paper, we go a step further and demonstrate via experimentation that packet-based TC algorithms such as [2] are reduced to statistical ML models and how a simpler state-of-the-art ML approach such as gradient boost can outperform a DL-based model.

III. Dataset and baseline DL model for experimentation

To demonstrate the interpretability problem in packet-based TCS that uses DL for performing the classification task, we will describe the dataset and DL approach used as the baseline for experimentation.

A. Dataset and Classification Tasks

For this research, we use ISCX VPN-nonVPN traffic dataset. This dataset contains real traffic from different applications which are stored in pcap format files [6]. The complete dataset comprises 14 traffic categories, 7 for a regular session (non VPN) and 7 over VPN. The traffic categories are Web Browsing, Email, Chat, Streaming, File Transfer, VoIP, and P2P. The subset of this dataset that was used for experimentation is that non-VPN traffic. This decision was taken to demonstrate how a DL model can be reduced to a simpler statistical IP/port-based/Packet Length ML model.

For simplicity and without loss of generality, we do not include the Web Browsing traffic class into the subset as different generated streams would belong to two categories (ex. video watched via web browsing). Each traffic category is composed of one or more categories of applications. The 16 applications used to label the data are: aim, email, Facebook, FTPS, Gmail, Hangout, ICQ, Netflix, SCP, SFTP, Skype, Spotify, Torrent, Vimeo, Voipbuster, Youtube. As a result of these two types of labels, the following traffic classification tasks are defined as in [2]:

- **Traffic characterization**: This task aims to discriminate between protocol families (e.g. chat or video).
- **Application identification**: This task aims to discriminate specific applications (e.g. Spotify or Skype).

Once the dataset ISCX VPN-nonVPN was obtained, we built a new dataset for our experiments as follows. We randomly select a maximum of 30K packets from each of the 110 pcap files (where possible). As the dataset is highly unbalanced, the next step was performing random under sampling. The resulting dataset has 24K packets per traffic category or 2.6K packets for each application category. The balanced dataset is then split into two subsets, 80% of the packets are used for training and validation, and 20% for testing. The unified training and validation dataset is further split to get

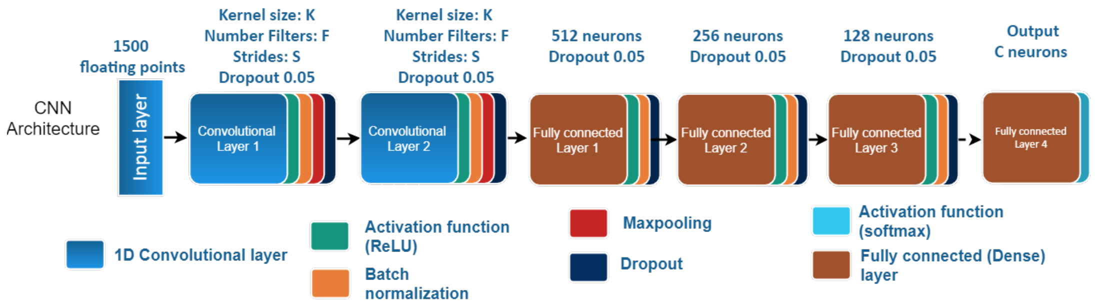


Fig. 1: Implemented CNN model to replicate the results of Deep Packet [2]

the individual training (85% of packets) and validation (15% of packets) datasets.

B. DL architecture and Packet Pre-processing

Depending on the type of DL architecture used to build the TC, several pre-processing steps can be implemented. For this paper, we perform the pre-processing steps proposed by Lotfollahi et al. [2]. Also, the DL models that we implement are inspired by their 1D-CNN architecture. Following the Deep Packet approach, we apply the following pre-processing steps to all packets in the dataset built for the experiments:

1) The Ethernet header is removed
2) All non-IPv4 or non-IPv6 packets are discarded
3) All non-TCP or non-UDP packets are discarded
4) All packets with no payload are discarded
5) The source and destination IP addresses are set to 0.0.0.0 (masked)
6) The IP header is padded to 40 bytes (IPv4 and IPv6 have different lengths)
7) The transport layer header is padded to 20 bytes (TCP and UPD have different lengths)
8) The payload is truncated or padded (with 0s) at 1440 bytes
9) The payload bytes are divided by 255 to produce a value in the range of [0,1] (normalization)
10) The whole processed packet is reassembled to a total size of 1500 bytes.

Figure 1 shows the implemented 1D-CNN architecture. This architecture is composed of two convolutional layers, followed by four dense ones. All the layers have rectifier activation functions (ReLU) except the last one with a soft-max function for classification. Max-poling is used in convolutional layers for down-sampling the input, while dropout and batch normalization layers were used in convolutional and dense layers to improve generalization and accelerate training. Two models were trained, one for traffic characterization and one for traffic identification, with the unique difference being the number of neurons in the last softmax layer (6 vs 16 classes). Table I presents the configuration of each Convolutional Layer. For training, a batch size of 128 was used, along with Adam as optimizer and categorical cross-entropy loss function. We implemented and trained the model using TensorFlow[1].

The results of the implemented model and the ones from Deep Packet [2] are compared in Tables II and III for Traffic Characterization and Application Identification, respectively. In general, the results of the implemented model are aligned to the results of Deep Packet except for the email class in the

TABLE I: 1D-CNN Model parameters

Task	Convolutional 1			Convolutional 2			Output Layer
	K	F	S	K	F	S	C
App Ident.	4	200	3	5	200	1	16
Traffic. Char.	5	200	3	4	200	3	6

TABLE II: Performance of implemented 1D-CNN model for traffic classification vs Deep Packet

	This paper			Deep Packet		
traffic	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
chat	0.78	0.92	0.85	0.84	0.71	0.77
email	0.63	0.95	0.76	0.96	0.87	0.91
ftp	0.99	0.97	0.98	0.98	1.00	0.99
p2p	0.72	0.99	0.84	1.00	1.00	1.00
streaming	0.96	0.93	0.95	0.92	0.87	0.90
voip	0.94	0.93	0.94	0.63	0.88	0.74

TABLE III: Performance of implemented 1D-CNN model for Application Identification vs Deep Packet

	This paper			Deep Packet		
App	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
aim	0.28	0.93	0.43	0.87	0.76	0.81
email	0.37	0.90	0.52	0.97	0.82	0.89
facebook	0.93	0.93	0.93	0.96	0.95	0.96
ftps	0.99	0.98	0.99	1.00	1.00	1.00
gmail	0.32	0.95	0.48	0.97	0.95	0.96
hangout	0.99	0.89	0.94	0.96	0.98	0.97
icq	0.34	0.71	0.46	0.72	0.80	0.76
netflix	0.99	0.97	0.98	1.00	1.00	1.00
scp	0.97	0.97	0.97	0.97	0.99	0.98
sftp	0.98	0.99	0.99	1.00	1.00	1.00
skype	0.98	0.81	0.89	0.94	0.99	0.97
spotify	0.80	0.92	0.86	0.98	0.98	0.98
torrent	0.65	0.97	0.78	1.00	1.00	1.00
vimeo	0.94	0.98	0.96	0.99	0.99	0.99
voipbuster	0.98	0.99	0.98	0.99	1.00	0.99
youtube	0.93	0.96	0.95	0.99	0.99	0.99

[1]https://www.tensorflow.org

Traffic Characterization task and the email, gmail, aim, and icq classes in the Application Identification task. We argue that the differences in the results may be due to 1) the datasets used for training/validation/testing (ours is more limited in size and there is a random component while selecting the packets to create the balanced dataset), 2) minor differences between the implemented model and Deep Packet, and 3) the stochastic nature of DL model optimizer. However, these differences do not have effects on the interpretability of the model.

C. Packet Assembly variations

To identify the effects of the raw input packets on the DL model is by modifying the pre-processing procedure and measuring the performance of the model trained with the new input format. As we described before, Deep Packet pre-processing is composed of 10 steps. In the last step, the processed **IP Header (IPH)**, **Transport Protocol Header (TH)**, and **Transport Protocol Payload (TPL)**, which carries the application data, are put back together to assemble the raw packet for classification. Each of these pieces of information carries on important information to discriminate between different classes of traffic. By modifying this assembling step, we target to provide some level of interpretability of the model and answers to questions such as: what is the model learning? How important is the header for learning from raw packets? Is the TPL important? What features are impacting the model?

In order to change the raw input data, We first define the following five strategies to assembly the packet:

- **DeepPacket**: No modifications to the approach followed by Deep Packet were applied. This is our baseline for the experiments.
- **noIP**: The input packet is only composed of the transport protocol data (header and payload). This modification will provide useful information about the usefulness of the IPH in the classification task.
- **payloadOnly**: Only the raw application data is used as input data. This modification will provide insights about the impact of the application data in the model's performance.
- **smallPayload**: Similar to DeepPacket but the TPL is shortened to 500 bytes. This experiment will complement the previous results.
- **headerOnly**: The IPH and TH are used. This experiment will show us if the model is being reduced to statistical IP/port-based architectures.

For each experiment, a DL model following the architecture showed in Figure 1 was created, trained, and their performance was evaluated in terms of F1-score.

IV. Performance Evaluation and Discussion

In this section, we will present the main results of this paper. We first analyse the performance impact of the different packet assembly variations described in the previous section. A data analysis process on the dataset is used to explain the results, and we further extend our experiments to demonstrate that simpler classifier can outperform the DL models. Finally, we complement the experiments by analysing the impact of the payload pre-processing on the results.

A. Packet assembly performance

Figure 2 shows the F1-Score achieved by the DL models trained with the raw input provided by the different packet assembly strategies on both traffic classification tasks. In both tasks, the headerOnly strategy, which uses the processed IPH and TH only, achieved similar or even better performance than the DeepPacket strategy. These results may indicate that the DL models are automatically extracting the features from the packet headers and that the payload information is not used to discriminate between classes.

By analysing the source and destination transport protocol pairs for Facebook and Vimeo traffic in Figure 3a, we can notice that they have different patterns, facilitating discrimination among different classes. Moreover, if we compare these patterns with the same traffic but from the TOR dataset, as shown in Figure 3b, we can see that this feature is useless for discriminating such traffic. This result may also explain the poor performance of Deep Packet on the TOR dataset. Finally, we can see that the performance is negatively impacted when less information from the header is used but we keep the TPL. We analyse the effects of the TPL in the next subsection.

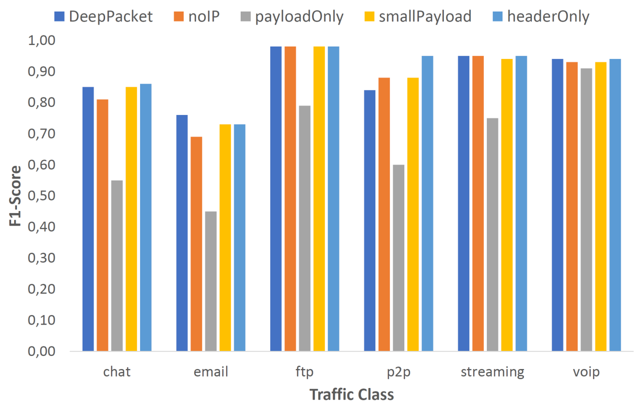
With these results, it seems that the DL models are reduced to a statistical IP/port-based architecture. To confirm this, we built and trained a simpler ML model based on Gradient Boost (GB) ensemble method, where we use as features the Transport Protocol pairs source-destination ports and its type (UDP/TCP). The results are presented in Figure 4. We implemented the GB algorithm using the LightGBM[2] framework.

In the Traffic Characterization task, the ML approach is outperformed by the DL ones, but their differences are always lower than 13 percentage points. Even in classes like email and p2p, the ML model performs better. On the contrary, the ML outperforms the DL approaches in most of the classes in the Application Identification task. The differences are even higher in the email classes, where the DL models were struggling. This result confirms the DL models are in fact a statistical transport protocol type and port-based architecture, which features are automatically extracted from raw (byte) representation of the packet headers.

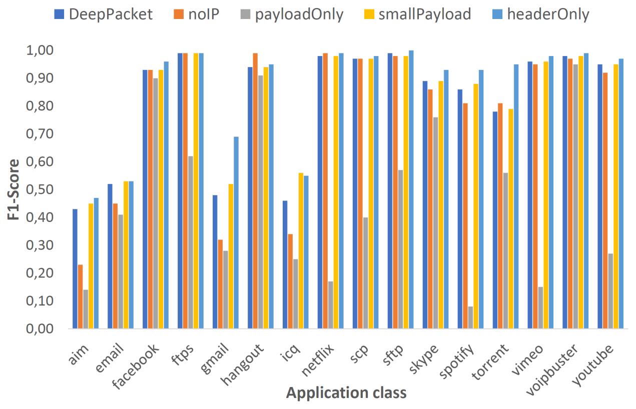
[2]https://lightgbm.readthedocs.io/

TABLE IV: Description of the different packet assembly strategies for different experiments

Experiment Label	Packet Assembly	Total Packet Size (Bytes)
DeepPacket	Deep Packet	1500
noIP	Transport Header and Payload	1460
payloadOnly	Transport Payload only	1440
smallPayload	IP and Transport Headers with small Payload (500 bytes)	560
headerOnly	IP and Transport Headers only	60

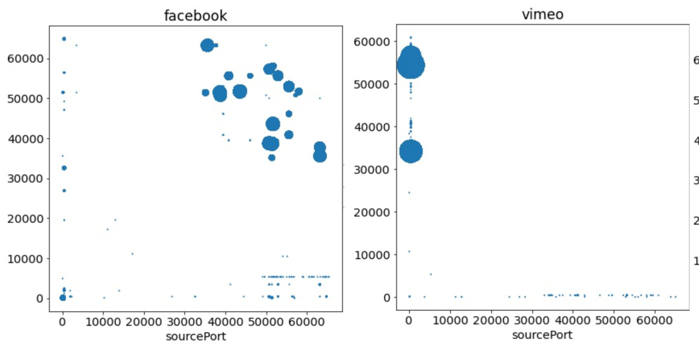


(a) Traffic Characterization

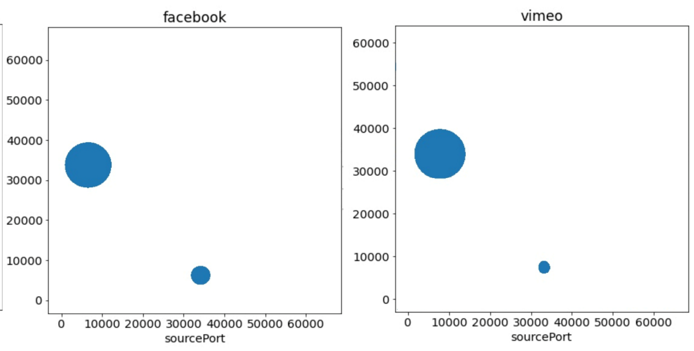


(b) Application Identification

Fig. 2: Comparison of the F1-Score achieved by the DL models in the different packet assembly variations. In both cases, headerOnly achieves similar or better performance than DeepPacket. On the contrary, payloadOnly had the worst performance.



(a) Source vs Destination Transport Protocol Port for non-VPN traffic



(b) Source vs Destination Transport Protocol Port for TOR traffic

Fig. 3: Comparison of source-destination Transport Protocol Port used by the Facebook and Vimeo traffic in the non-VPN and TOR datasets. While non-VPN traffic has a evident source-destination port pair patterns, this is not the case for TOR traffic.

B. Payload impact

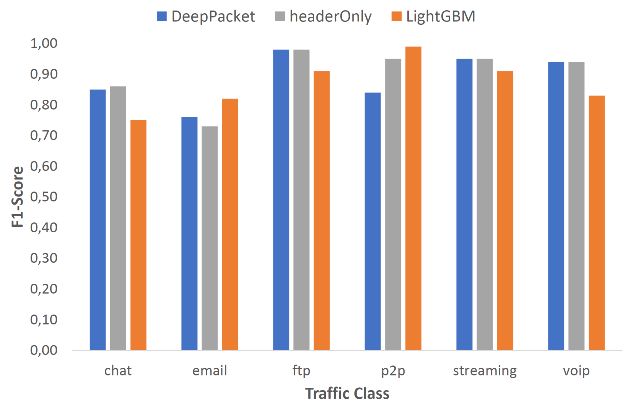
One interesting aspect of the previous results is that the TPLs are not providing features to improve the DL models' performance, and it may be even negatively impacting it. We performed some additional experiments to identify the importance of the TPL in DL models. In the first experiment, we changed the eight step of the packet pre-processing by padding the packets with a length shorter than 1440 bytes with random sequences. With this experiment, we want to identify if the DL models are automatically extracting the length of the payload as a feature. Due to paper length constraints, we only present the Traffic Characterization task results, but we found similar results for the other task.

Figure 5a shows the payloadOnly experiment's performance when using the zero padding, as in the previous experiments, and random padding. Notice that when the padding is random, the DL approach cannot perform the classification task correctly, as in p2p classification where the performance drops up to 90%. This can be explained as the packet length can quickly emerge from the zero padding pattern. Of course, this feature will only help when the different traffics have different packet lengths. In the used dataset, a data analysis process showed us that each traffic packet length distribution is a distinctive feature among several classes.

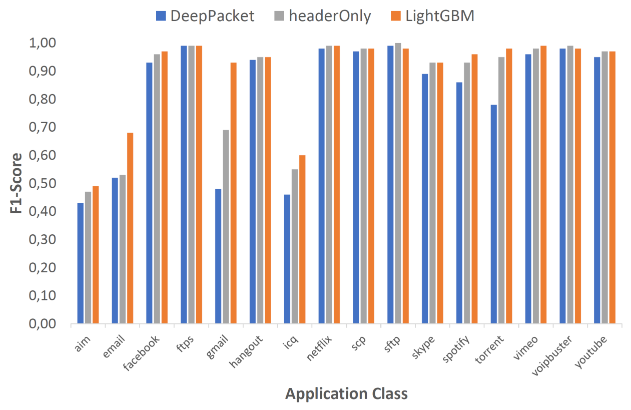
To enforce our understanding of the packet length's impact, we built an ML model using LightGBM and trained it using the original packet length as a feature. Figure 5b shows the comparison between headerOnly, onlyPayload and the model using LightGBM using only packet length as the only feature. Although headerOnly DL model still outperforms the other two models, payloadOnly and LightGBM are having similar performance. This may also explain why DeepPacket still has a little advantage over headerOnly in the previous results. In general, we can see how a combination of data analysis, feature engineering and domain knowledge allow building ML models with high interpretability, high accuracy, and low computational complexity for non-encrypted packets. Features like packet lengths, transport protocol type and source-destination port pairs are enough to solve the traffic classification tasks.

V. Conclusions and Future work

Motivate by the lack of interpretability of Deep Learning (DL) models for packet-based Traffic Classifier (TC) on raw

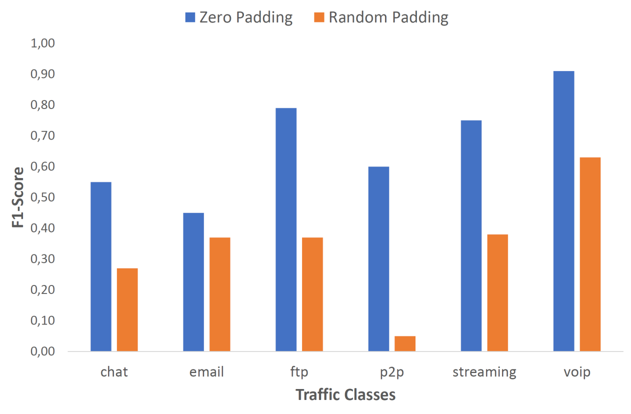


(a) Traffic Characterization

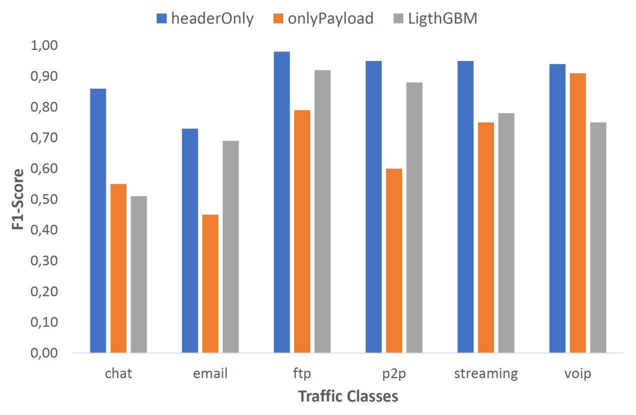


(b) Application Identification

Fig. 4: Comparison of DeepPacket and headerOnly DL models against an ML based on Gradient Boost. Although the ML model is slightly outperformed in Traffic Characterization by the DL models, it outperformed both in Application Identification.



(a) payloadOnly packet assembly with two different padding strategies



(b) Comparison between two of the DL models and one using statistical ML-based with packet length as a unique feature.

Fig. 5: Interpreting the impact of packet length in the DL models. By comparing the result of padding strategies during packet pre-processing, we see that this feature is being learned and we validate it via a simpler ML model.

input data, this paper presented a set of experiments to measure the prediction accuracy of the DL-based TCs when different combinations of headers and payloads from the byte representation of the packet are used to train them. Based on the evaluation results, we were able to reveal what features the DL models are automatically extracting and learning and validate it by building a simpler Machine Learning (ML)-based TC that could outperform the DL approach. As future work, we plan to apply more advanced techniques for analyzing model interpretability on more complex datasets, e.g., on VPN traffic, and use other DLML models to generalize the results.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101017109 "DAEMON".

References

[1] F. Pacheco, E. Exposito, M. Gineste *et al.*, "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.

[2] M. Lotfollahi, R. Shirali hossein zade, M. Jafari Siavoshani *et al.*, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, 09 2017.

[3] F. Shaheen, B. Verma, and M. Asafuddoula, "Impact of automatic feature extraction in deep learning architecture," in *2016 International conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–8.

[4] G. Aceto, A. Montieri, A. Pescapè *et al.*, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. PP, 02 2019.

[5] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," 2018.

[6] A. Habibi Lashkari, G. Draper Gil, M. Mamun *et al.*, "Characterization of encrypted and vpn traffic using time-related features," 02 2016.

[7] W. Wang, M. Zhu, J. Wang *et al.*, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," 07 2017, pp. 43–48.

[8] R. Moreira, L. F. Rodrigues, P. F. Rosa *et al.*, "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.