Network Traffic Classification for Detecting Multi-Activity Situations

Ahcene Boumhand, Kamal Singh, Yassine Hadjadj-Aoul, Matthieu Liewig, César Viho

Network Traffic Classification for Detecting Multi-activity Situations

Ahcene Boumhand[+], Kamal Singh[*], Yassine Hadjadj-Aoul[$], Matthieu Liewig[+], and César Viho[$]
[+]*Orange Labs, Rennes, France*
firstname.lastname@orange.com
[*]*Univ Jean Monnet, IOGS, CNRS, UMR 5516, LaHC, F - 42023 Saint-Etienne, France*
firstname.lastname@univ-st-etienne.fr
[$]*Univ Rennes, Inria, CNRS, IRISA, Rennes, France*
firstname.lastname@irisa.fr

***Abstract*—Network traffic classification is an active research field that acts as an enabler for various applications in network management and cybersecurity. Numerous studies from this field have targeted the case of classifying network traffic into a set of single-activities (e.g., chatting, streaming). However, the proliferation of internet services and devices has led to the emergence of new consuming patterns such as multi-tasking that consists in performing several activities simultaneously. Recognizing the occurrence of such multi-activity situations may help service providers to design quality of service solutions that better fit users' requirements. In this paper, we propose a framework that is able to recognize multi-activity situations based on network traces. Our experiments showed that our solution is able to achieve promising results despite the complexity of the task that we target. Indeed, the obtained multi-activity detection performance is equivalent or often surpasses state-of-the-art techniques dealing with only a single activity.**

***Index Terms*—Network traffic classification, multi-activity situations, machine learning**

I. Introduction

Network traffic classification is a widely studied field with a broad spectrum of applications in network management such as assuring quality of service (QoS) and resource planning and allocation. Numerous studies [1] have focused on classifying network traffic into different activities according to different levels of granularity such as the traffic type (e.g., chatting) and the application type (e.g., Skype). However, these studies considered only the case when a single activity (e.g., chatting, or mailing, or streaming, etc.) is performed at a time.

In this paper, we target the specific case of multi-activity situations that we define as being composed of two or several simultaneous activities (e.g., mailing and streaming and etc.). To our knowledge, our current study is the first to treat such a concept as a network traffic classification task. As a matter of fact, with the proliferation of devices, services, and applications, multi-activity situations are occurring more frequently in our daily routines. Therefore, recognizing such complex situations will provide new ways for service providers to analyse such composite patterns in real time. This will help to better understand the service and user requirements and to better adapt their network management solutions. Moreover, the concept of multi-activity has been the subject of many recent studies [2], [3] (particularly in the field of social sciences) which have shown its widespread occurrence among various communities of the population (e.g., adolescents, teleworkers) and its potential impact on health and productivity.

In order to tackle the problem of multi-activity situations detection, we propose a solution that allows a time window of a network trace to be associated with a specific multi-activity class. This solution consists in the two following main contributions.

- At present, there exists only single-activity network traces public datasets. Thus, we propose a new method that generates multi-activity network traces starting from single-activity network traces.
- To recognize the occurrence of multi-activity situations, we conceive a classifier that pre-processes a multi-activity network trace and then assigns it into a predicted multi-activity class. Additionally, as we base our classifier on machine learning techniques, we conduct a comparison between the performance of some recent models (e.g., LSTM with attention mechanisms, Transformer).

Classifying multi-activity network traces based on temporal windows introduces new challenges that harden the network traffic classification task. In fact, the interference of bidirectional flows that belong to several activities as well as noise flows creates a composite network traffic behavior that differs significantly from single-activity network traffic behavior.

The remainder of this paper is organized as follows. Section II reviews a set of selected works that are related to network traffic classification. In Section III, we present the main components of our multi-activity detection methodology. Section IV describes the experimental parameters that we leveraged to instantiate our multi-activity detection methodology. Section V depicts the performance of our solution together with a comparison with some related works. In Section VI, some of the potential improvements and technical choices regarding our methodology are discussed. Finally, Section VII concludes the paper while giving the main directions for future work.

II. Related work

In order to address the problem of network traffic classification, researchers and practitioners usually leveraged traditional techniques such as port-based methods and deep packet inspection (DPI). However, as the properties of network traffic evolved, the performance of those techniques was highly impacted. Port-based methods assume a fixed mapping between the ports utilized by the applications and the registered port numbers by IANA (Internet Assigned Numbers Authority). Such methods have become inefficient because the modern applications use dynamic port numbers and thus any fixed mapping is becoming rare. On the other hand, DPI techniques base their approach on inspecting packets' content in order to search for specific patterns and signatures. Users' privacy concerns and the adoption of encryption techniques have significantly affected the usefulness and effectiveness of DPI techniques.

In this context, the usage of classical machine learning (ML) techniques have emerged as one of the alternatives for performing network traffic classification tasks. Network traffic classification techniques that are based on classical ML models (e.g., support vector machine, decision trees, random forest) are able to achieve a high level of accuracy [4], [5] by leveraging statistical characteristics of network flows (e.g., mean packet inter-arrival time) and without requiring to inspect packets' payloads. Posterior to that, the major recent successes of Deep Learning (DL) models in other research domains have inspired their adoption in the network traffic classification field [6], [7]. Indeed, DL models are known for their ability to extract features from data automatically and their high capacity to learn and model complex patterns. These abilities allow DL models to generally surpass other classical ML techniques.

Network traffic classification can be leveraged to infer users' activities according to three levels of granularity: traffic type (e.g., chatting, streaming), application type (e.g., Skype, Instagram), and the behaviors that are performed within a specific application (e.g., send a message on WhatsApp, post a tweet on Twitter). For the first level of granularity (*traffic type*), to differentiate between a set of traffic types (e.g., mailing, VOIP), authors in [8] used a combination of K-means and random forest (RF) over a set of statistical features (e.g., packet count send-receive ratio, mean time between packets' arrival). Whereas in [9], authors proposed an attention aided long short-term memory (LSTM) as well as a hierarchical attention network (HAN) that receives a sequence of packets' payloads. A similar approach was proposed in [10] by leveraging a two-dimensional convolutional neural network (CNN) and a LSTM.

For the second level of granularity (*application type*), and to distinguish between a given set of applications, authors in [11] based their method on a Markov chain and a RF models to analyse both the sequential behavior and the statistical characteristics (e.g., packet size distribution) of a network flow. In [12], multi-head attention was leveraged over a sequence of packets' payloads and metadata (e.g., packet position, packet size), while in [7] a combination of gated recurrent units (GRU) and one-dimensional CNN were leveraged over a set of features quite similar to those in [12].

Finally, for the third level of granularity (*user behavior*), the work in [13] focuses on classifying multiple user behaviors within the Instagram application by using a support vector machine (SVM) model. Some examples of user behaviors in this context are login into the service or posting a video. For a wider range of applications (e.g., browsing potential matches on Tinder, posting to Facebook), the authors in [14] based their work on a combination of an unsupervised algorithm K-means and a supervised algorithm SVM. For a similar purpose to [14], the authors in [15] based their work on a combination of agglomerative hierarchical clustering and RF.

Despite their high level of achieved accuracy, one common point in the previously cited works is that they treat only the case when a single activity is performed at a time sequentially. These techniques are therefore not capable to detect multi-activity situations. Furthermore, single activities do not cover all real-world scenarios where a user may perform two or several activities simultaneously (e.g., chatting with a friend on WhatsApp while listening to a podcast on Spotify). These are some of the reasons that led us to investigate the detection of multi-activity situations.

III. Multi-activity detection methodology

The aim of this section is to provide a thorough overview of the methodology that we adopted to build a solution that is able to recognize multi-activity situations based on network traces and ML techniques.

The overall architecture of our solution is composed of two chained processes. In section III-A, we describe the first process that enables us to create a multi-activity dataset starting from a dataset of single-activity network traces. Different steps are required to generate a multi-activity dataset. First, we select repetitively network traces that belong to two different activities, and then we merge them in order to generate a multi-activity network trace. When applied systematically, this process generates a complete multi-activity dataset. This process is described in Figure 1. In section III-B, we depict the second process of the architecture by listing its different components that allow us to preprocess and then assign a given multi-activity network trace into a predicted multi-activity class (see Figure 2).

A. Multi-activity network traces dataset

1) Dataset of single-activity network traces: In the following, we consider the single-activity dataset as a set of N network traces (i.e., raw pcap files) $D = \{T_1, T_2, T_3, \ldots, T_N\}$ where each network trace is an ordered sequence of packets. We suppose that each network trace is associated with only one activity (label) that refers to the activity (e.g., mailing, VoIP) that was performed when the corresponding trace was captured. Furthermore, we define a set of functions that enable us to access some properties of a network trace or perform an operation on single or multiple traces:

- `duration`(T_i): this function indicates the duration of a network trace T_i
- `random`(min, max): this function draws a real number uniformly from the interval [min, max]
- `duplicate`(T_i, d): this function creates duplicates of the trace T_i until it reaches the duration d, then it returns the duplicates as a set of traces
- `RanDistr`(S, d): distributes randomly a set of traces S over a duration d. This produces a trace of duration d that gathers the traces that are contained within S and separates them with random no-activity durations.
- `merge`(T_i, T_j): this function produces an output trace that gathers the packets that are incoming from the two input traces and merge while sorting them according to their arrival time.

2) Generation of multi-activity network traces: First, we define a multi-activity situation as the performance of two or several activities of different types simultaneously (e.g., mailing and streaming). Our current study focuses on the specific case of two simultaneous activities, which corresponds to the maximum number of activities studied in social sciences [2], [3]. Multi-activity situations of a superior order will be considered in future work.

Our goal is to generate a multi-activity network trace starting from two single-activity network traces T_i and T_j where we assume that T_i has a longer duration than T_j, as described in Algorithm 1. To reach this goal, we first generate a random multi-activity duration (MA_D) that indicates the duration during which the activities corresponding to T_i and T_j are supposed to be performed simultaneously and that does not exceed the duration of T_i ($0 < MA_D <$ `duration`(T_i)) (see line 1). Posterior to that, if the generated multi-activity duration is bigger than T_j, we duplicate T_j until it reaches the desired multi-activity duration (MA_D). These duplicates are gathered in a set named S (line 2). Then, the duplicates of T_j are spread along the duration of T_i and are separated by random durations of no-activity phases. This will create a trace T'_j that has the same duration as T_i and that alternates between single-activity phases (i.e., phases of the activity that corresponds to T_j) and no-activity phases (line 3). It is important to notice that contrary to the timestamps of the packets of T_i that are kept the same during the merging operation, the timestamps of the packets of T_j and its duplicates may receive a delay to be shifted to the convenient position in T'_j.

In the final step, packets from the traces T_i and T'_j are gathered in the output trace T_{out} and reordered according to their arrival time (i.e., timestamp) (line 4).

B. Multi-activity network traces classification

Herein, we describe thoroughly the steps that are involved in preprocessing following with assigning a multi-activity network trace to a predicted output class. This process is shown in Figure 2.

1) Hypotheses and requirements: The proposed classification must respect the following requirements:

- The system must be able to recognize multi-activity situations the most accurately possible.
- The system must be able to recognize these situations within 10 seconds.

These competing requirements dictate us to find a balance between reaction time and providing the classifier with enough information so it can perform its predictions accurately.

2) Fragmentation and labelling process: In this process, incoming traces from the multi-activity network traces dataset will undergo a fragmentation operation as a primary step for preparing an adequate input for the classifier. Each multi-activity network trace is split into time windows of W seconds. This time window is a hyper-parameter that has to be chosen wisely in a manner that enables us to satisfy the constraints described previously. Specifically, the selected window size has to fall within a range of values that are large enough to provide the classifier with a sufficient amount of knowledge to perform its inferences correctly. Moreover, the selected window size has to be small enough to allow our proposed framework to provide responses with a delay that is shorter than 10 seconds.

3) Features extraction: Features or inputs of a ML model are one of the main parameters that control the quality of the trained model predictions. Indeed, a well-picked set of features may not only improve the performance of the ML model but can also play a prominent role in reducing the training time, the risk of over-fitting, and the size of the ML model.

At this level, fragments that are issued from the previous process are represented as time series of L time steps where each time step is a sub time-window of size $\frac{W}{L}$ seconds. This representation will allow us to have a fine-grained description of each fragment as the features will not be computed over the entire window but over each sub time-window.

Statistics over packet sizes and inter-arrival times are widely used in the network traffic classification field and have shown their usefulness in differentiating between traffic types in numerous works like in [5], [16]. Thus, in our work, the following features are computed over the packets that are contained within each time step of the time series: mean, variance, skewness, kurtosis of packet size and packet inter-arrival time, the sum of packet size, and packets' count.

4) Classifier: Since we base our solution on ML based algorithms, the classifier can behave in two modes: (i) the training mode where it receives the set of features associated

Algorithm 1 `gen-multi-activity-trace`(T_i, T_j)

1: $MA_D \leftarrow$ `random`$(0,$ `duration`$(T_i))$ ▷ generate the multi-activity duration
2: $S \leftarrow$ `duplicate`(T_j, MA_D) ▷ duplicate T_j until it reaches MA_D
3: $T'_j \leftarrow$ `RanDistr`$(S,$ `duration`$(T_i))$ ▷ spread the duplicates of T_j along the duration of T_i
4: $T_{out} \leftarrow$ `merge`(T_i, T'_j) ▷ gather then reorder the packets of T_i and T'_j in the trace T_{out}

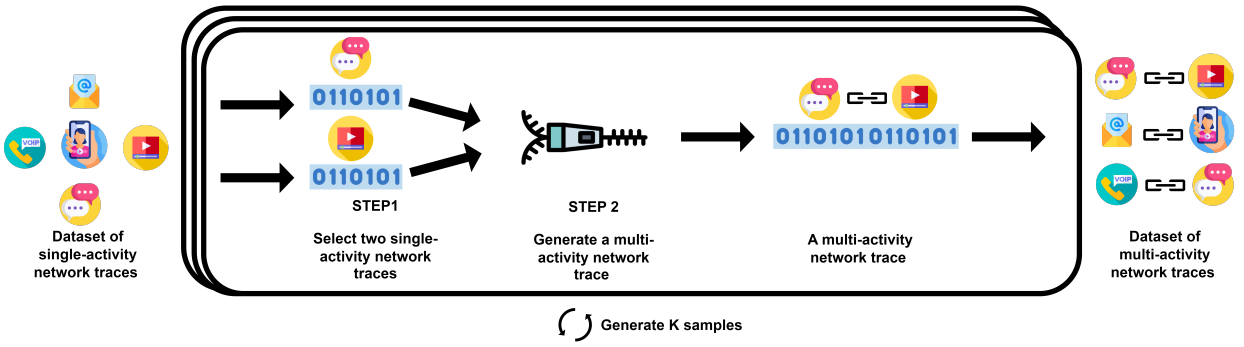

Fig. 1. Process of creating a synthetic multi-activity dataset from a single-activity dataset. (Images from freepik.com and flaticon.com)

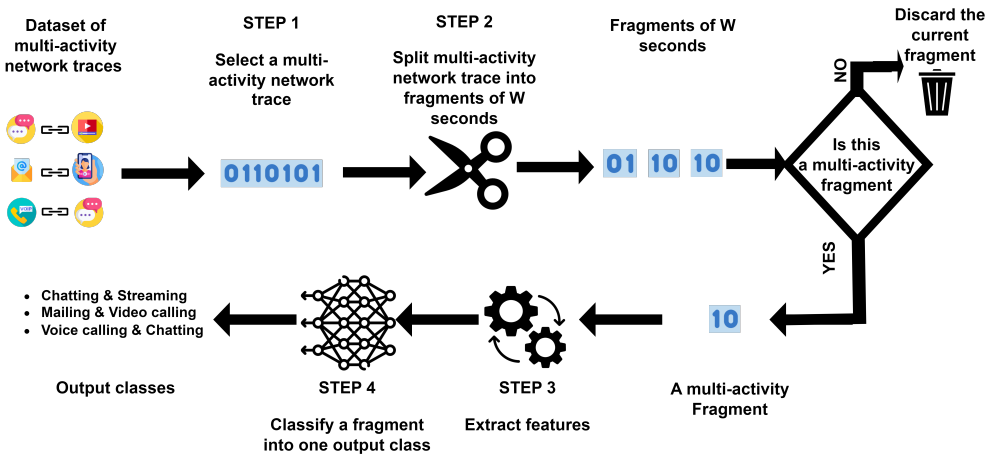

Fig. 2. Process of training a multi-activity classifier. (Images from freepik.com and flaticon.com)

with a fragment along with its label to tune its parameters, and (ii) the inference mode where the classifier can be fed with a set of features associated with a fragment and infer its predicted multi-activity class.

IV. Experimental Setup

In this section, we depict the resources, tools, and hyper-parameters that we selected to instantiate the methodology that has been described in Section III.

To generate synthetic multi-activity network traces, we leveraged the ISCXVPN2016 dataset [5] as a source for single-activity network traces. This dataset is publicly available and has been used in several previous works to train and test their traffic classification models. It can also serve as a common base to compare our solution to other related works. The ISCXVPN2016 dataset is constituted of a set of network traces (28 GB of raw pcap traces) that are labeled according to the particular activity that was performed when the trace was captured along with the application within which the activity was performed (audio call on Skype, audio streaming on Spotify). Due to the storage and computational complexity of the multi-activity traces generation process, we restrained in the current study our single-activity network traces selection on the following set of activities and applications: Chat (Facebook, Hangouts), Email, Streaming (Spotify, Youtube), VoIP (Skype, VoIPBuster). The application of the multi-activity traces generation process on this selected set of single-activity traces yielded into a multi-activity dataset whose traces are labeled with one of the following six labels: Chat & Mailing (C1), Chat & Streaming (C2), Chat & VoIP (C3), Mailing & Streaming (C4), Mailing & VoIP (C5), and Streaming & VoIP (C6). The resulting dataset was then split into three sets: the training set (60%), the validation set (20%), and the test set (20%).

For the multi-activity traces classification process, we set the configuration parameters W and L to 10 and 40, respectively (see section VI). This means that the classifier receives as input a time window of 10 seconds that is represented as time series (i.e., segmented into sub-time windows) of 40 timesteps where each timestep is constituted of the set of features that are described in section III-B3 and that are computed over the 0.25 (i.e., $\frac{10}{40}$) seconds related to this timestep.

To implement the classifier, we leveraged a set of two ML models that are RF and XGBoost, and three DL models

that are Bi-LSTM with attention mechanisms, 1D-CNN with attention mechanisms and a Transformer model.

During the hyper-parameters tuning process, we set both ML models (i.e., RF and XGBoost) to the default hyper-parameters. Specifically, in the implementation we leveraged, the number of estimators is set to 100 for both models, while the maximum depth is set to 6 for XGBoost and without limit for RF. On the other side, the following hyper-parameters were adopted to tune the three deep-learning models. For the LSTM model, we used a Bidirectional LSTM with 10 units in each direction and an enabled return-sequences option. The recurrent layer was then enhanced using an additive attention mechanism [17] and followed by a dense layer with 12 units which is twice the number of output classes. For the CNN model, we utilized a one-dimensional CNN layer with 40 filters, and a kernel size and stride that were set to 1. This architecture was adopted based on a comparison that we conducted between a model with one CNN layer and a model with two CNN layers where the former showed better results on the validation set. Similarly to the recurrent DL model, the CNN layer was enhanced using an additive attention mechanism [17] and followed by a dense layer with 8 units. Finally, for the Transformer model, we harnessed one block of a transformer's encoder [18] where we kept the default architecture (e.g., layer normalization, residual connections) while we set the multi-head attention hyper-parameter number of heads (to 4) and the key dimension (to 10). To determine the number of blocks in the transformer's encoder, we varied this hyper-parameter within an interval that ranges from 1 to 6. The models with 1 and 5 blocks achieved similar results while outperforming the rest of the models. Hence, we selected the model with the lowest number of parameters. The transformer's encoder was then followed by a layer of one-dimensional global average pooling, a dropout layer with a rate that was set to 0.1, and a dense layer with 8 units. The hyper-parameters' values of the three DL models were tuned by testing various combinations of values for the different hyper-parameters and then selecting the ones that optimized the results on the validation set. It is worth noting that for the three DL architectures, we leveraged ReLU as an activation function for all dense layers except for the last layer which consisted of 6 units with SOFTMAX as an activation function. Moreover, each dense layer except the last layer was followed by a dropout layer with a rate that was set to 0.1. In order to prepare the data for the DL models, we used normalization to scale the features to the range $[-1, 1]$. In addition, we leveraged the up-sampling technique to mitigate the impact of the imbalanced multi-activity dataset by preventing the model from inclining towards majority classes. Finally, to train the DL models, we leveraged the Adam optimization algorithm as an optimizer and used a batch size of 64. Moreover, we used the early stopping technique to avoid overfitting.

V. Evaluation and comparison

The performance of our proposed framework is measured using the following standard ML metrics: *accuracy, precision, recall, macro average F1 score,* and *weighted average F1 score*. The ML and DL models are implemented using Scikit-learn, TensorFlow 2, and Keras, and the tests are run on a PC with an Intel i7 CPU, NVIDIA Quadro RTX 3000 GPU, and 32 GB of RAM.

A. Comparison between models

The aim of this part is to compare the performance of the set of ML and DL models that we selected to instantiate the classifier component of the multi-activity network traces classification process.

TABLE I
Performance of multi-activity situations classification with different ML algorithms

Model	Accuracy	M-average F1	W-average F1
XGBoost	**0.83**	**0.81**	**0.83**
Random forest	0.80	0.76	0.80
Transformer	0.80	0.78	0.80
Bi-LSTM with attention	0.72	0.69	0.73
1D-CNN with attention	0.71	0.65	0.71

Table I gives the values of accuracy, macro-average F1 score, and weighted average F1 score that are reached by the models XGBoost, RF, Bi-LSTM with attention, 1D-CNN with attention, and Transformer. These results show that XGBoost outperforms the other models by reaching an accuracy of 0.83. The XGBoost model is succeeded by the Transformer and the RF models that attain an accuracy of 0.80. Lastly, Bi-LSTM with attention and 1D-CNN with attention achieves the worst results with an accuracy of less than 0.72. The M-average and W-average F1 scores show that our up-sampling method correctly creates a balanced dataset.

These results may be explained on one hand by the nature of the input data that are tabular and that resulted from a feature engineering process. This type of data is more suited for classical ML models than DL models [19]. Hence, tree-based ensemble models (XGBoost, RF) performed better than the rest of the models. On the other hand, these results are consistent with the recent studies [18] that showed the supremacy of Transformer models over attention-aided recurrent networks.

B. Detailed analysis of the XGBoost model performance

In order to investigate the performance of the best-performing model from section V-A, we summarize in Table II the results that are achieved by the XGBoost model for each of the possible classes when tested on the test set. These results

TABLE II
Detailed performance results of XGBoost classifier for multi-activity situations

Class	Precision	Recall	F1 score	Support
C1 - Chat & Mailing	0.72	0.86	0.78	539
C2 - Chat & Streaming	0.96	0.87	0.91	1780
C3 - Chat & VoIP	0.89	0.75	0.81	6398
C4 - Mailing & Streaming	0.83	0.70	0.76	682
C5 - Mailing & VoIP	0.63	0.93	0.76	2023
C6 - Streaming & VoIP	0.85	0.87	0.86	9955

are presented using the metrics *Precision, Recall*, and *F1 score* along with the *Support* that indicates the number of instances of each class in the test set. The reported results highlight that the XGBoost model is able to differentiate between the different classes with a satisfactory level of robustness. Indeed, the value of the F1 score is above 0.75 for all the classes. It surpasses the threshold of 0.80 for three classes (i.e., Chat & Streaming (C2), Chat & VoIP (C3), Streaming & VoIP (C6)), and exceeds the value of 0.90 for one class (i.e., Chat & Streaming (C2)).

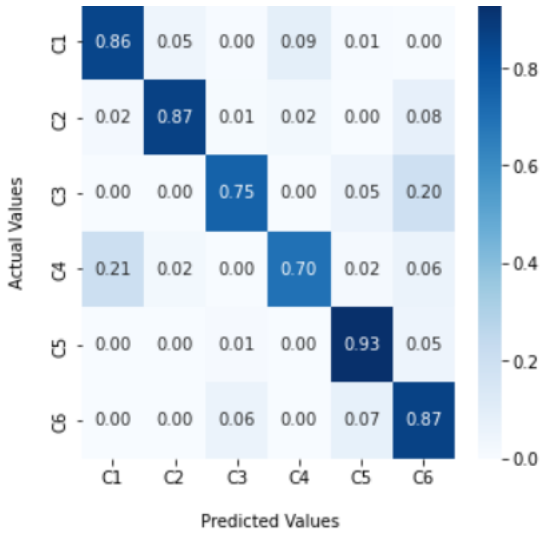

Fig. 3. Confusion matrix of XGBoost model classification on 6 classes of multi-activity situations.

To deepen our knowledge of the performance of the XGBoost model and gain a detailed and accurate understanding of its classification process, we provide in Figure 3 the confusion matrix related to the XGBoost model. The rows of the confusion matrix correspond to the actual class of the instances, while the columns correspond to the predicted class of the instances. This figure reveals the absence of misclassification patterns in our model for all the classes except for two classes (i.e., Chat & VoIP (C3) and Mailing & Streaming (C4)) as the misclassified samples are distributed over all the non-actual classes. For the classes Chat & VoIP (C3) and Mailing & Streaming (C4), we can observe that around 20% of the misclassified instances are attributed to the classes Streaming & VoIP (C6) and Chat & Mailing (C1), respectively. It is important to notice that in both cases at least one of the activities that compose the multi-activity situation was predicted correctly (i.e., VoIP in Chat & VoIP (C3), and Mailing in Mailing & Streaming (C4)).

C. Comparison with related works

In order to compare the performance of our model against the state-of-the-art, we selected a set of baselines from the network traffic classification field that harnessed the ISCXVPN2016 dataset [5] to classify network traffic into activities. This set of baselines are described in Table III that summarizes for each work the way with which it segments a network trace to prepare the input for the classification model (i.e., classification unit in the 2nd column), the ML model it uses to perform the classification (3rd column), along with the output classes and the results it achieves.

As it can be seen in the last column of Table III, the existing related works consider only single-activity classes (e.g., Chat, VoIP). To the best of our knowledge, our work is the only work that targets the case of multi-activity classes (e.g., Chat & VoIP, Chat & Streaming).

The column Results from Table III shows that even though our framework targets a more complex and harder task (i.e., multi-activity situations classification), it is able to achieve comparable results to those related to only single-activity reported in [20] and [16]. Lastly, the work in [8] exhibits better performances. However, some necessary information that allow us to compute the weighted F1 score are lacking and it targets only four single activity classes.

VI. Discussion

In this section, we discuss some of the technical choices that we adopted in our methodology in order to expose their alternatives that we can explore in future works.

In our experimental setup, we opted for a value of 10 seconds as a time window size when we fragment the multi-activity network traces to prepare the input for the classifier. This value was chosen heuristically to allow the classifier to perform its inferences accurately while allowing the system to react promptly upon users' actions. Nevertheless, this parameter could have been selected in a more elaborate manner by varying the window size within a predefined range and then selecting the value that allows the classifier to attain the best results.

In the fragmentation and labeling process, we labeled a fragment of a network trace as a multi-activity fragment if it contained at least two packets, one belonging to each activity. This adopted labeling technique may produce multi-activity instances that are the hardest to classify (i.e., instances with only one packet for one or both activities). This constraint could have been softened by considering only fragments that contain more than a certain predefined number of packets belonging to each activity as a multi-activity instance.

To create the training dataset, we applied our proposed multi-activity traces generation process (see section III-A2) based on the single-activity network traces that are provided in the ISCXVPN2016 dataset [5]. Hence, the resulting multi-activity network traces can be described as synthetic or artificial. However, in order to evaluate the performance of our framework when tested in a real environment, real network traces of multi-activity situations must be collected. This can be the purpose of our future studies.

VII. Conclusion

In this paper, we presented a framework that is able to recognize multi-activity situations based on network traces. We believe that such a solution may endow operators and service

TABLE III
COMPARISON WITH SOME RELATED WORKS THAT ARE DEDICATED TO ACTIVITY DETECTION ON ISCXVPN2016 DATASET [5]

Reference	Classification unit	Classification model	Output classes	Results	Multi-activity detection
[20]	Bidirectional flow	GRU, 1D CNN, MTL	6 classes: VoIP, File Transfer, P2P, Streaming, Chat, Email	Accuracy: 0.81 F1 score: 0.79	×
[16]	Bidirectional flow	1D CNN, MTL	5 classes: Chat, Email, File transfer, Streaming, VoIP	Accuracy: 0.81 F1 score: non available	×
[8]	Time window	K means, RF	4 classes: Video, Chat, Bulk, VoIP	Accuracy: non available F1 score: 0.90	×
Our work	Time window	XGBoost	6 classes: Chat & Mailing, Chat & Streaming, Chat & VoIP, Mailing & Streaming, Mailing & VoIP, Streaming & VoIP	Accuracy: 0.83 F1 score: 0.83	✓

providers with new insights about analysing users' modern consumption patterns and requirements, and hence, enables the operators to better adapt their network management solutions. Our framework is constituted of a process that generates synthetic multi-activity network traces from single-activity network traces. A classifier then assigns the input multi-activity network traces to the corresponding multi-activity class. We provided an instance of our proposed methodology leveraging the publicly available dataset ISCXVPN2016 [5] and a set of ML and DL models.

To evaluate the performance of our solution, we conducted a set of experiments on the test set that showed that the XGBoost model outperformed the rest of the implemented ML and DL models. The obtained results showed also that our solution is able to achieve good results even if our framework targets a more complex and harder task (i.e., multi-activity situations classification) as compared to existing related work that only consider single-activity.

As future research directions, we intend to test our methodology on other datasets to validate its consistency and performance. Moreover, it may be of great interest to refine the output labels in order to provide not only the simultaneously performed activities (e.g., Chatting & Streaming) but also the applications on which the multi-activities are performed (e.g., Chatting on Skype & Streaming on YouTube). Finally, to harness the full potential of DL models, it can be relevant to test other types of features that favor the usage of end-to-end DL.

REFERENCES

[1] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning–based approaches for internet traffic classification," *Annals of Telecommunications*, vol. 75, pp. 673–710, 2020.

[2] E. Beuckels, G. Ye, L. Hudders, and V. Cauberghe, "Media multitasking: A bibliometric approach and literature review," *Frontiers in psychology*, vol. 12, 2021.

[3] E. Ophir, C. Nass, and A. D. Wagner, "Cognitive control in media multitaskers," *Proceedings of the National Academy of Sciences*, vol. 106, no. 37, pp. 15583–15587, 2009.

[4] S. Sengupta, N. Ganguly, P. De, and S. Chakraborty, "Exploiting diversity in android tls implementations for mobile app traffic classification," in *The World Wide Web Conference*, pp. 1657–1668, 2019.

[5] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pp. 407–414, 2016.

[6] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[7] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer networks*, vol. 165, 2019.

[8] V. Labayen, E. Magaña, D. Morató, and M. Izal, "Online classification of user activities using machine learning on network traffic," *Computer Networks*, vol. 181, 2020.

[9] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 241–252, 2019.

[10] Z. Zou, J. Ge, H. Zheng, Y. Wu, C. Han, and Z. Yao, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 329–334, IEEE, 2018.

[11] C. Xiang, Q. Chen, M. Xue, and H. Zhu, "Appclassifier: automated app inference on encrypted traffic via meta data analysis," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2018.

[12] J. Cheng, R. He, E. Yuepeng, Y. Wu, J. You, and T. Li, "Real-time encrypted traffic classification via lightweight neural networks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[13] H. Wu, Q. Wu, G. Cheng, S. Guo, X. Hu, and S. Yan, "Sfim: Identify user behavior based on stable features," *Peer-to-Peer Networking and Applications*, vol. 14, no. 6, pp. 3674–3687, 2021.

[14] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, vol. 16, pp. 69–78, 2016.

[15] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2015.

[16] S. Rezaei and X. Liu, "Multitask learning for network traffic classification," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, IEEE, 2020.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2022.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, 2021.