Flow Feature-Based Network Traffic Classification Using Machine Learning

Nicolas A. T. de Menezes and Flávio L. de Mello

***Abstract* — Reliable network traffic classification is essential to management and security tasks. Therefore, it is beneficial to analyze and improve existing techniques. Some of the most traditional methodologies for traffic classification are based on port number and packet payload, each of which presents an increasing set of problems. Port number-based classification techniques suffer from the misuse of port numbers and tunneling. This is primarily due to their reliance on the proper use of IANA (Internet Assigned Numbers Authority) assigned numbers. On the other hand, packet payload-based classification has difficulty dealing with encrypted data and legal restrictions to accessing user data. Flow feature-based classification can overcome these challenges by creating profiles based on the traffic patterns of applications. Furthermore, machine learning techniques have shown to be a good match for traffic classification. Thus, the goal of this paper is to explore the combination of these fields and to develop a set of machine learning models capable of classifying network traffic based on flow features. This was achieved by using a ready to use dataset to train two supervised and one unsupervised clustering model. The results for the supervised classifiers were considered comparable to similar studies, while the performance of the clustering model was found to be not satisfactory.**

***Index Terms* — Computer networks, Network traffic classification, Machine learning**

I. Introduction

TRAFFIC classification can have many vital applications in the management and security of computer networks, such as traffic prioritization, performance monitoring and anomaly detection. This classification task can be summarized as the mapping of the incoming traffic into classes of interest in a reliable and accurate manner.

Traditional classification techniques that rely on port numbers have been shown to no longer be reliable enough [1, 2, 3], primarily due to the misuse of port numbers in order to obfuscate traffic [4]. Moreover, the widespread use of cryptography has posed problems for more robust classification techniques, such as payload-based DPI (Deep Packet Inspection) [5, 6].

Considering these challenges, flow feature-based classification has shown to be a potential alternative to port number and packet payload-based techniques. Furthermore, machine learning can be an ideal tool to automatically find a mapping between the traffic flows and the classes of interest [4]. Thus, the main goal of this work is to develop a set of machine learning models capable of achieving a classification accuracy comparable of that present in the literature.

The general approach adopted by this work consists of: 1) identifying common software tools, machine learning models and practices for building datasets in the relevant literature; 2) selecting a set of machine learning models to train and evaluate; 3) executing the classification experiments on the trained models and evaluating the results.

This paper is organized as follows: section II presents some of the related works. Section III explains the setup used for the experiments and discusses in detail the adopted dataset. Section IV presents and analyses the results of each experiment. Finally, section V presents some conclusions and suggests future works.

II. Related Work

The basis of this work was presented by Boutaba *et al.* [4], who performed a comprehensive survey of the applications of machine learning in computer networking, such as resource management, congestion control and network security. A particular focus was given to the studies related to traffic classification, where the authors presented three classification methodologies: 1) payload-based; 2) host behavior-based; 3) flow feature-based.

Roughan *et al.* [8] compared the performance between k-NN and LDA (Linear Discriminant Analysis) to classify traffic into QoS (Quality of Service) classes of service, such as *Interactive*, *Bulk* and *Streaming*. The authors created a dataset from proprietary data and used flow features such as the average packet size and average flow duration. They managed to obtain an error rate of 5.1% and 9.4% for k-NN using 4 and 7 classes, respectively. They also found that the error rate between specific classes, such as *Streaming* and *Bulk*, is much higher than the average, indicating that better features needed to be derived for these classes.

The study conducted by Moore and Zuev [12] analyzed the performance of a NB (Naïve Bayes) model for classifying classes of applications, such as *WWW*, *MAIL* and *BULK*. The authors used a dataset developed in an earlier work [2, 7] in conjunction with kernel density estimation and FCBF (Fast Correlation-Based Filtering) in order to improve the model's accuracy. They found that the accuracy increased from 65.26% for no processing to 93.50% and 94.29% for the

Nicolas A. T. de Menezes, Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, menezes.nicolas@poli.ufrj.br.

Flávio L. de Mello, Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, fmello@poli.ufrj.br.

datasets treated with kernel density estimation and FCBF, respectively. The authors also used a second dataset, containing data collected 12 months later and using the same features as the original, for evaluating the time robustness of the model. They found a much lower accuracy of 20.75% for the unprocessed dataset and 37.65% when kernel density estimation was applied. They also found that the use of FCBF increased the accuracy to 93.38%, an indicator that the fewer features present in this version of the dataset didn't suffer many variations.

Alternatively, Park, Tyan and Kuo [9] expanded the work made by Moore and Zuev [12] and compared the performance and computational cost between a decision tree and a Naïve Bayes model. They also used QoS classes of service, like those in Roughan *et al.* [8], but restricted the flow features to those that could be easily collected or computed from common network monitoring tools used by ISPs (Internet Service Providers). Some of the features used were the size of the packets, the duration of the flow and the packet inter-arrival time. The authors found that the DT (Decision Tree) presented very similar results to the NB model but with significantly lower computational complexity.

Erman, Mahanti and Arlitt [10] compared the performance between AutoClass [15], K-Means and DBSCAN. They used two datasets, a publicly available packet trace called Auckland IV [13] and a proprietary dataset collected from the University of Calgary's internet access link. Each dataset contained features derived from transport layer statistics and each flow was assigned an application class, used for labeling and evaluating the resulting clusters. This assignment was done using a port-based classification for the Auckland IV set, while a payload-based method [14] was adopted for their proprietary dataset. The average accuracy for the Auckland IV dataset was found to be 79% for K-Means with K at around 100, 92.4% for AutoClass and 75.6% for DBSCAN using 3 *minPts* and 0.02 *eps*.

III. Traffic Classification Experiments

This section details the adopted methodology for performing the experiments. Subsection A explains the software tools and the hardware that were used. The adopted datasets are explained in more detail in section B. Finally, section C discusses the preprocessing steps taken before training the models.

A. Environment

The main software tool used in this study for preprocessing the dataset, training and evaluating the models was Weka [16] version 3.8.5, developed in Java by the machine learning team at the University of Waikato. The Explorer view of the program was used in this work because it allows for quick testing of a variety of models.

Three algorithms were considered in this experiment: 1) the supervised k-NN (k-nearest neighbors) model; 2) the supervised J4.8 decision tree; 3) the unsupervised DBSCAN (density-based spatial clustering of applications with noise). All the experiments were performed on a Windows 11 PC with 16 GB or memory and an AMD hexacore processor running at 4.00 GHz.

TABLE I
Number of Flows per Class

Class	Number of Flows in the Main Dataset	Number of Flows in the Evaluation Dataset
WWW	328,092	15,597
MAIL	28,567	1,799
FTP-CONTROL	3,054	289
FTP-PASV	2,688	695
ATTACK	1,793	0
P2P	2,094	297
DATABASE	2,648	295
FTP-DATA	5,797	529
MULTIMEDIA	576	0
SERVICES	2,099	121
INTERACTIVE	110	4
GAMES	8	0
TOTAL	377,526	19,626

Number of flows per class in the main and evaluation datasets created by [2] and [7].

B. Datasets

This study used two datasets containing 248 features and 12 classes defined by Moore and Papagiannaki [2] and Moore, Zuev and Crogan [7]. Both sets contained data collected form a single, full-duplex Gigabit Ethernet link connecting over 1,000 personnel at a set of biology-related laboratories to the internet. Table I shows the number of flows per class for both datasets.

The first and main dataset contained information from a continuous 24-hour capture from September of 2003, resulting in a total of 377,526 flows. While the authors divided this data into 10 subsets, for the purposes of this study all the subsets were merged back together.

The second dataset contained data collected 12 months later, from September of 2004. The resulting number of flows was much lower, at around 19,626, due to a storage space limitation at the packet capture equipment. Following the methodology presented by Moore and Papagiannaki [2], this dataset was used to evaluate the time robustness of the classifier. Therefore, it was named as the evaluation set.

Despite initially considering building our own dataset from network traces, we faced some difficulties with the NetMate [17] software that made settling for a ready-to-use dataset the better option. This software is used for extracting flows and computing their features and is available at GitHub [18]. A series of bugs, a lack of usable output and the fact that the project hadn't been maintained in over 11 years made us abandon this approach.

C. Preprocessing

Some machine learning models, such as k-NN, can be sensitive to feature scaling and non-numeric attributes. Therefore, some processing of the datasets is required before it can be used for training. The same treatment was given to both the training and the evaluation sets, with some small

TABLE II
NUMBER OF FLOWS PER CLASS

Class	Number of Flows in the Main Dataset	Number of Flows in the Evaluation Dataset
WWW	1,000	500
MAIL	1,000	500
FTP-CONTROL	1,000	289
FTP-PASV	1,000	500
ATTACK	1,000	0
P2P	1,000	297
DATABASE	1,000	295
FTP-DATA	1,000	500
MULTIMEDIA	576	0
SERVICES	1,000	121
INTERACTIVE	0	0
GAMES	0	0
TOTAL	9,576	3,002

Number of flows per class in processed versions of the main and evaluation datasets used for the experiments.

TABLE III
PERCENTAGE OF CLASSIFIED FLOWS PER CLUSTER

Class	Largest Cluster	Second Largest Cluster	Noise
WWW	39.5%	37.7%	12.9%
MAIL	76.4%	4.1%	6.7%
FTP-CONTROL	93.5%	0.7%	0.9%
FTP-PASV	79.1%	0.7%	5.1%
ATTACK	87.6%	0.4%	6.1%
P2P	56.1%	18.4%	7.4%
DATABASE	93.2%	0.0%	2.1%
FTP-DATA	90.3%	0.1%	3.6%
MULTIMEDIA	74.1%	0.0%	5.6%
SERVICES	98.8%	0.0%	1.2%
TOTAL	79.1%	6.5%	5.3%

Percentage of class flows in each of the two largest clusters and noise generated by the DBSCAN model.

adjustments made to account for technical differences in the ARFF files. All preprocessing steps were performed using one of Weka's built-in filter classes. Table II shows the number of flows for each dataset used in the training and evaluation of the models.

Firstly, a subsampling step was applied using the *weka.filters.supervised.instance.SpreadSubsample* filter. This was done to prevent an imbalanced dataset affecting the performance of the models, since 86.9% of all samples in the main dataset belonged to the *WWW* class. Furthermore, the sample ratio between the *WWW* and the second largest class, *MAIL*, was 11.5. Therefore, for the main dataset, at most 1,000 flows were randomly selected per class. This value was chosen to be 500 for the evaluation set to account for the fewer number of samples.

Secondly, the classes *INTERACTIVE* and *GAMES* were dropped from the main dataset due to insufficient samples (only 110 and 8 flows, respectively). This was done using the *weka.filters.unsupervised.instance.RemoveWithValues* filter. The evaluation dataset already had no *GAMES* samples, so only the *INTERACTIVE* class was removed.

The final steps involved scaling and treating the categorical (non-numeric) features of both datasets. The former involved transforming each feature to obtain 0 mean and a standard deviation of 1. This was performed using the filter *weka.filters.unsupervised.attribute.Standardize*. All categorical features present in the datasets were binary (Y or N). Therefore, the latter consisted of mapping an N into 0 and a Y into 1.

IV. RESULTS

A. DBSCAN Clustering

The overall accuracy obtained for the main dataset using *minPts* equal to 6 and *eps* equal to 1.2 was of only 23.93%. For this reason, the evaluation dataset was not used in this model. In total, 21 clusters were created, out of which 5.3% of all the flows were classified as noise and 79.1% of flows were bundled into a single cluster. Table III shows the distribution of class samples into the two largest clusters and noise.

Due to time and processing constraints, a proper selection of the optimal input parameters couldn't be performed. While the clustering time took only 71 seconds for *eps* equal to 1.2, it increased well above one hour for smaller values without producing noticeable improvements. Therefore, the final value had to be randomly selected within reason, which turned out to produce unsatisfactory results. Our analysis indicates that better performance could be achieved in this dataset with a parametrization scheme like that in Erman, Mahanti and Arlitt [10].

B. k-NN Classifier

The k-NN model has only one input parameter, K, which was chosen by first training a set of models in the main dataset with K between 1 and 10 and then picking the one with the highest accuracy. The final value of K used was 5 and the results for both datasets are shown in Table IV.

This approach was taken, as opposed to selecting K as the number of classes in the dataset, due to the higher accuracy of the resulting model. Another factor was that Weka already provided a built-in mechanism for this selection, therefore making it trivial to experiment with.

For the main dataset, this model performed very well with an overall accuracy of 90.87%. Despite this, the confusion matrix indicated that 9.8% of the flows in the WWW class were incorrectly classified as ATTACK, while 11.1% of the ATTACK samples were mistaken for WWW. These numbers were significantly higher than the other classes and accounted for most of the accuracy loss. According to [8], this could be due to the high similarities between the flows in these two classes.

For the evaluation dataset, the overall accuracy dropped to 62.69%. This result was expected, and our analysis suggests a few factors: 1) the main dataset did not contain enough data to allow the model to generalize well. This could be primarily due to the data originating from a single continuous 24-hour trace; 2) the feature distribution changed due to new applications appearing or changing in the span of 12 months;

TABLE IV
PERFORMANCE METRICS FOR THE K-NN MODEL

Class	Accuracy	Precision	Recall	F1-score
Main Dataset	90.87%	90.8%	90.9%	0.908
Evaluation Dataset	62.69%	77.3%	62.7%	0.658

Overall accuracy, precision, recall and F1-score of the k-NN model on the main and evaluation datasets.

TABLE V
PERFORMANCE METRICS FOR THE J4.8 DECISION TREE

Class	Accuracy	Precision	Recall	F1-score
Main Dataset	97.35%	97.4%	97.3%	0.973
Evaluation Dataset	41.67%	NaN	93.9%	NaN

Overall accuracy, precision, recall and F1-score of J4.8 decision tree on the main and evaluation datasets.

3) the presence of redundant or irrelevant features could be negatively affecting the performance of the model.

It is important to note that, by reducing the number of features to 12 on the same datasets using FCBF, Moore and Zuev [12] managed to maintain a very similar accuracy on the evaluation data. This could be a strong indicator for point 3) presented above.

Interestingly, the recall of the three FTP classes (FTP-CONTROL, FTP-PASV and FTP-DATA), rated at 82.7%, 69.4% and 97.2%, respectively, indicated that these flows did not suffer from the possible presence of extra features.

C. J4.8 Decision Tree

The decision tree algorithm was trained and then pruned using the reduced error pruning method. This method takes a bottom-up approach and replaces nodes with their respective most common classes until this results in an accuracy loss. While not the most effective, this method is fast and allowed for a quicker evaluation. The resulting tree had a depth of 15 and contained 96 nodes. Table V shows the detailed results for both the main and the evaluation datasets. Note that some of the metrics could not be computed by Weka for the evaluation dataset and, therefore, appear as NaN in the table.

On the main dataset, the tree had a very high overall accuracy of 97.35%. Like the k-NN model, the confusion matrix indicated that the highest class mix-up occurred between WWW and ATTACK, with 5.2% of WWW samples being mistaken for ATTACK and 7.0% vice-versa.

The results for the evaluation dataset showed an overall accuracy of 41.67%, which was expected according to the analysis performed on the k-NN model. Like the k-NN model, the three FTP classes retained very high recall values, all above 90%.

V. CONCLUSION

The main goal of this work is to reproduce the general methodology for training a machine learning model capable of flow feature-based traffic classification. In this regard, the results obtained for the k-NN classifier and the J4.8 decision tree show comparable results with similar works [7, 8, 9], including the low classification accuracy for the evaluation dataset. However, the performance of the DBSCAN model shows a disparity from the results in Erman, Mahanti and Arlitt [10], which requires further evaluation.

This work also faced some challenges during the construction of a dataset from network traces, which lead to the use of a ready to use dataset [2, 11]. This caused the loss of control over the represented data, which was found to be lacking in variety causing the poor classification performance on the evaluation set.

A. Future Work

Future works can be suggested based on some of the limitations and challenges presented by this paper, namely: 1) the further evaluation and training of a clustering model capable of achieving accuracy comparable to that in Erman, Mahanti and Arlitt [10]; 2) the construction of a dataset containing more representative data in order to address the limitations with the adopted dataset; 3) the use of RNNs (Recurring Neural Networks) to address the issue of temporal robustness of the model.

REFERENCES

[1] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy and M. Faloutsos, "Is p2p dying or just hiding?", in GLOBECOM, vol. 3, pp. 1532-1538, 2004, DOI: 10.1109/GLOCOM.2004.1378239

[2] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications", in PAM, vol 5, pp 41-45, 2005, DOI: 10.1007/978-3-540-31966-5_4

[3] H. Dreger, A. Feldmann, M. Man, V. Paxson and R. Sommer, "Dynamic application-layer protocol analysis for network intrusion detection", in USENIX Security Symposium, pp 257-272

[4] R. Boutaba *et al*., "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities", Journal of Internet Services and Applications, may, 2018, DOI: https://doi.org/10.1186/s13174-018-0087-2

[5] L. Bernaille and R. Teixeira, "Implementation issues of early application identification", Lecture Notes in Computer Science, 4866:156, 2007, DOI: 10.1007/978-3-540-76809-8_14

[6] J. Erman, A. Mahanti, M. Arlitt and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core", in Proceedings of the 16th international conference on World Wide Web, ACM, pp; 883–892, DOI: https://doi.org/10.1145/1242572.1242692

[7] A. W. Moore, D. Zuev and M. Crogan, "Discriminators for use in flow-based classification", Queen Mary University of London, aug., 2005

[8] M. Roughan *et al*., "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification", in Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, pp. 135–148, 2004, DOI: https://doi.org/10.1145/1028788.1028805

[9] J. Park, H. Tyan and C. Kuo, "Internet traffic classification for scalable qos provision", in IEEE International Conference on Multimedia and Expo, pp. 1221–1224, 2006, DOI: 10.1109/ICME.2006.262757

[10] J. Erman, A. Mahanti and M. Arlitt, "Traffic Classification Using Clustering Algorithms", in SIGCOMM Workshops, sept., 2006, DOI: https://doi.org/10.1145/1162678.1162679

[11] A. W. Moore *et al*., "Architecture of a Network Monitor", Passive & Active Measurement Workshop, 2003

[12] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques", in Proc. ACM Sigmetrics, Alberta, Canada, pp. 50–59, Jun. 2005, DOI: https://doi.org/10.1145/1064212.1064220

[13] WAND Group, "WITS: Auckland IV", Available: https://wand.net.nz/wits/auck/4/auckland_iv.php, Accessed on: Feb. 13, 2022.

[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINK: Multilevel Traffic Classification in the Dark", in *SIGCOMM'05*, Philadelphia, USA, Aug. 21-26, 2005, DOI: https://doi.org/10.1145/1080091.1080119
[15] C. P. Cheeseman *et al.*, "AutoClass: a Bayesian classification system", in Proceedings of the Fifth International Conference on Machine Learning, pp. 54–64, 1988, DOI: https://doi.org/10.1016/B978-0-934613-64-4.50011-6
[16] E. Frank, M. A. Hall, I. H. Witten, "The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques", 4 ed.: Morgan Kaufmann, 2016
[17] S. Zander and C. Schmoll, "NetMate - User and Developer Manual", unpublished, 2004
[18] D. Arndt, "GitHub: DanielArndt/netmate-flowcalc", Available: https://github.com/DanielArndt/netmate-flowcalc, Accessed on: Feb. 12, 2022

Nicolas A. T. de Menezes is an undergraduate student in Electronics and Computer Engineering at UFRJ. He works at Siemens Energy as a software developer since 2021. He was also a student at the LPS lab at UFRJ and worked on a project involving power electronics.

Flávio Luis de Mello received his DSc. in Theory of Computation and Image Processing from the Federal University of Rio de Janeiro - UFRJ (2006), MSc. in Computer Graphics from the Federal University of Rio de Janeiro - UFRJ (2003), Undergraduate degree in Systems Engineering from the Military Institute of Engineering - IME (1998).

He developed command and control systems and implemented military messages interchange applications during twelve years as a Brazilian Army officer. He was responsible for developing software applications based on machine learning and knowledge reasoning from Mentor Group.

Dr Mello currently is Associate Professor at the Electronic and Computer Engineering Department (DEL) of Polytechnic School (Poli) at the Federal University of Rio de Janeiro (UFRJ). He is head of the Machine Intelligence and Computing Models Laboratory (IM2C).