# Traffic Classification with Machine Learning in a Live Network

Jarrod Bakker, Bryan Ng, Winston K.G. Seah, and Adrian Pekar

***Abstract*—This paper reports on our experience with deploying network traffic classifiers in a live Software Defined Network (SDN). We select five simple machine learning (ML) algorithms and implement them for Distributed Denial of Service (DDoS) detection. Using publicly available datasets, we establish a standard reference for the performance of each classifier (algorithm) in terms of accuracy, precision and detection rate. An identical experiment over a live SDN shows that the classifiers perform significantly poorer compared to the reference standard, exhibiting up to 11.2% lower accuracy, 30.2% lower precision and detection rate lower than 15% (98% in the reference standard). We argue that the interactions between network elements such as the switch and the controller significantly significantly affects the performance of ML algorithms in a live network which must be accounted for in a real deployment.**

***Index Terms*—Traffic classification, machine learning, SDN, nmeta**

## I. Introduction

Network security plays a critical role in computer networks. It is a highly specialised and technical discipline that covers intrusion detection and prevention systems, identifying vulnerabilities, exploits, and threat mitigation. Over the past decade, network security and attacks have advanced rapidly, yet, the evolution of network attacks have far outpaced the security responses [1], [2]. Smarter and more efficient approaches must be developed to keep networks and data secure in this era of rapid evolution of network attacks.

A technique that is continuously increasing in popularity is machine learning (ML) based traffic classification (TC). TC describes the process of identifying and pairing of packet flows to traffic types and this process relies on information extracted from the traffic that serves as input for TC. The goal of TC is to improve the management of network resources, network security, and the Quality of Service (QoS) [3], [4], [5].

Traffic classification mechanisms classify traffic based on packets or flows as information passes through the network. Packet-based classification (also known as deep packet inspection) uses information extracted from both, the packet headers as well as the packet payloads. Flow-based classification collects packets into flow records and stores aggregated information such as the number of bytes and packets per flow. ML has significant advantages over payload inspection based techniques. For instance, payload inspection examines the payload of each packet, encryption, however, defeats this approach [6], [7]. Proponents of ML also argue that ML techniques involve much lower computational overheads than deep packet inspection techniques [8].

978-3-903176-15-7 

ML is the next step in the evolution of traffic classification as it looks beyond these features. As such, the popularity of ML in recent years has seen the coupling of flow statistics with ML approaches to classify network traffic [6]. Statistical classification uses an underlying probability model to calculate the probability of a case belonging to a class. Given a case, its class can be determined through knowledge of its attributes. Background knowledge can be used to inform the classification process and human intervention can be used to modify the variables as needed.

Most of the state-of-the-art ML-based TC approaches claim high accuracy and performance [9], [10], [11]. They achieve these results in a static, offline environment. We argue, that, as we show in this paper, the performance of ML-based traffic classifiers substantially drops when deployed over live networks. A live network in the context of this paper is defined as a running network that serves user needs within an organisation. As new user devices, new Internet services and applications appear across an organisation's network, the traffic becomes more heterogeneous while continuously expands in both volume and speed. In consequence, making operational sense out of the diverse, ever-growing volume of network traffic is a major challenge.

Achieving reliable TC with ML remains poorly studied especially in a live network. This paper reports on our experience gained when using ML methods for TC over a live SDN. The contributions of this paper are as follows:

1) An investigation into the application of statistical classifiers for DDoS attack detection.
2) Evaluation results that have been obtained from a physical network testbed as opposed to a virtualised environment.

The remainder of the paper is organised as follows: Section II lists related work. Section III describes our methodology and the data that is used in the experiment. Section IV discusses the obtained results while Section V summarises the learnt lessons. Finally, Section VI concludes our work.

## II. Related Work

A promising approach to achieve reliable and accurate network anomaly detection is the combination of flow statistics with ML. Yet, only a small number of works is aimed at live networks.

Thapngam *et al*. [12] applied Linear Discriminant Analysis (LDA) to detect DDoS attacks on web servers. The classification was performed on the web server itself by sampling the arrival rate of incoming traffic. Pearson's Correlation

Coefficient and Shannon Entropy (information entropy) were used to calculating the dependency and predictability of traffic. The calculated values were used as features for classification. LDA generates a linear hyperplane to separate classes within an $n$-dimension coordinate system using Bayes' Theorem [13], [14]. As a result, the features being used for classification must contain continuous data. LDA addresses binary classification problems by using a single hyperplane to divide the coordinate system into two areas.

Roughan *et al*. [15] evaluated various Quadratic Discriminant Analysis (QDA) classifiers to TC for QoS policy enforcement. They evaluated the QDA-based classifiers alongside LDA and k-nearest neighbour (kNN)-based classifiers by using the same sets of features. Interesting enough, they discovered that the QDA-based classifiers performed worse compared to the other classifiers. As with any classification problem, these results are particular to the problem being solved and are not reflective of the QDA method as a whole.

Similarly to LDA, QDA uses a hyperplane within an n-dimension coordinate system to separate classes. The hyperplane that is generated however is described using a quadratic function. QDA is preferred over LDA when the covariance matrices for the two classes are not equal [14]. The use of a quadratic decision boundary makes this method more flexible than LDA whilst still being able to support continuous data.

The Support Vector Machine (SVM) method can be viewed as an improvement on the LDA and QDA methods mentioned previously [14]. These improvements relate to the positioning of the hyperplane and the ability to more efficiently operate in high dimensional spaces. The position of the hyperplane affects the classifier's ability to handle cases where the values for the corresponding features are close to the decision boundary, these points can also be referred to as noise or boundary cases. Bias can be introduced if the hyperplane is too close to the training data points belonging to one class, this may result in the misclassification of noise.

The performance of a SVM classifier is also dependent on the dimensionality of the model being created. The dimensionality of the coordinate system used in the model is equal to the number of features being used, similarly to LDA and QDA. Therefore, it follows that a training set with $n$ features results in a $n$-dimensional model. Increasing the number of features increases the complexity of calculating an optimal separating hyperplane.

The hyperplane-based methods explored above suffer from a lack of flexibility. Regardless of the function used to generate the decision boundary, the fixed nature of the boundary means that cases may be misclassified if they appear to be too similar to cases of other classes. In comparison, the simple principle used by kNN is effective in situations where the decision boundary contains irregularities [16].

The kNN method is possibly one of the easier classification methods to understand. This is in part due to its simplicity whereby unseen cases are classified based on the class of neighbouring cases [17]. The classification process consists of two steps: (i) find $k$ training instances that are closest to the unseen case, and (ii) map the most commonly occurring class from the $k$ neighbours as the class for the unseen case.

## III. Method and Data

The approach we have adopted in our research is closely related to an online vs. offline comparison. We first establish a reference standard performance for five classifiers (Random Forest, QDA, Naive Bayes, kNN and SVM) using publicly available network traffic datasets as inputs and compare it with the performance over a live network. In this paper, the reference standard is defined as the best achievable performance of a given classifier when the complete network traffic dataset is co-located with the machine executing the algorithm.

We design a set of experiments replaying the same network traffic datasets used to establish the reference standard on a live network and compare the performance of the three classifiers to that of the reference standard. Setting up the experiments, test bed and implementing the classifiers for an SDN has helped us gaining insights and experience which forms the contributions outlined in Section I.

We use two datasets, namely the "Center for Applied Internet Data Analysis DDoS Attack 2007 dataset" (hereafter, just CAIDA dataset) and the "University of New Brunswick ISCX Intrusion Detection dataset" (hereafter, just ISCX dataset). The CAIDA dataset is a packet trace with malicious and non-malicious flows from a DDoS attack while the ISCX dataset contains traces composed of HTTP denial of service (DoS), HTTP GET DDoS attack and a brute force SSH attack. The network data in the CAIDA dataset was captured over a 1 hour period, measures 21 GB in size while the ISCX dataset was captured over a 24 hour period, is 24 GB in size with a total number of $34,983,042$ packets recorded.

There is no guarantee that non-malicious flows are not present in the CAIDA dataset trace. Thus, we use background traffic from Victoria University's network (cleaned and curated) in place of the non-malicious flows to establish the reference standard. The ISCX dataset is supplied with a PCAP file containing both malicious and non-malicious flows, as well as a set of XML files that provides metadata on each flow. The metadata allows each flow to be identified via a five-tuple while provides some statistical information on the flow and allows us to establish the reference standard.

### A. Features

Table I lists the individual features used to build the feature sets. The *source* refers to the host that sent the first packet within the flow; typically the client. The *destination* is the recipient of the first packet; typically the server. We assume that the packets or bytes sent by the source are received by the destination and vice versa.

Six feature sets were composed using the features from Table I. Each feature set combines the different statistical properties of network traffic (as seen in Table I) to form a different set of predictors to be used by the classification methods. The feature sets were:

1) `totalSourceBytes` and `totalSourcePackets`
2) `totalSourceBytes` and `totalDestinationBytes`
3) `totalSourceBytes` and `flowDuration`
4) `totalSourceBytes` and log(`flowDuration`)
5) log(`totalSourceBytes`) and `flowDuration`
6) `destinationBytes-per-Packet` and `sourceBytes-per-Packet`.

TABLE I
FEATURES USED TO BUILD THE FEATURE SETS FOR THE FIRST GENERATION CLASSIFIERS.

| Feature | Description |
|---|---|
| `totalSourceBytes` | The number of bytes sent by the source. |
| log(`totalSourceBytes`) | The base 10 logarithm of `totalSourceBytes`. |
| `totalDestinationBytes` | The number of bytes sent by the destination. |
| `totalSourcePackets` | The number of packets sent by the source. |
| `flowDuration` | The duration of a bidirectional flow in seconds. |
| log(`flowDuration`) | The base 10 logarithm of `flowDuration`. |
| `sourceBytes-per-Packet` | The number of bytes sent from the source divided by the number of packets sent by the source. |
| `destinationBytes-per-Packet` | The number of bytes sent from the destination divided by the number of packets sent by the destination. |

### *B. Reference Standard*

The reference standard in this paper uses five metrics for each classifier. Let $A$ and $A'$ denote a classification (e.g. malicious), then each reference standard metric is expressed as a function of the confusion matrix (i.e. True Positive, True Negative, False Positive and False Negative) defined as follows:

*True Positive:* The actual class of a case was $A$ and the predicted class was $A$. This represents a successful prediction.

*True Negative:* The actual class of a case was $A'$ and the predicted class was $A'$. This represents a successful prediction.

*False Positive:* The actual class of a case was $A'$ and the predicted class was $A$. This represents an unsuccessful prediction.

*False Negative:* The actual class of a case was $A$ and the predicted class was $A'$. This represents an unsuccessful prediction.

The true positive rate (or recall) is the ratio of successful predictions made to cases of class $A$. This will be referred to as the detection rate (DR) throughout the rest of the paper as positive predictions will concern the detection of malicious traffic and is expressed as follows:

$$DR = \frac{TP}{TP + FN}. \tag{1}$$

The false positive rate (FPR) is the ratio of unsuccessful predictions made to cases of class $A'$:

$$FPR = \frac{FP}{TN + FP}. \tag{2}$$

Accuracy is the ratio of successful predictions made to both classes an expressed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3}$$

Precision (or positive predictive value) is the ratio of correct predictions made for class $A$ and is shown in Equation (4):

$$Precision = \frac{TP}{TP + FP}. \tag{4}$$

The f-measure statistic (or F1 score) considers both the DR and precision of a classifier to measure its quality:

$$\text{f-measure} = 2 \times \frac{DR \times Precision}{DR + Precision}. \tag{5}$$

The standard reference is evaluated on 75% of data (from the entire dataset) for training, 15% for testing and 10% for validation. The evaluation results are given in Table II.

### *C. Live network experiment*

The testbed used for the evaluation was built using a hardware-based environment. Figure 1 illustrates the network topology and provides a high-level overview of the traffic sent between each device. The network can be sliced into two parts. The first slice was used for network traffic sent over the data-plane and the second for the control-plane and test control network.

The data-plane consists of three hosts and a switch. A host was configured to replay network traffic (traffic source), a host configured to receive the traffic (sink), a host configured to run the nmeta2 DPAE (Data Plane Axillary Engine) [18] application and an OpenFlow v1.3 compliant Allied Telesis switch (AT-x930-28GSTX).

We use the nmeta2 framework to run the experiments. The nmeta2 framework consists of two separate applications. The first application, simply called nmeta2, is a Ryu controller application that manages OpenFlow v1.3 compliant switches. The second application, called nmeta2dpae, is designed to run on a dedicated machine that is separate from the controller and perform traffic classification on bidirectional flows. A bidirectional flow extends the notion of a unidirectional flow — a flow consisting of traffic sent from one host to another — by considering the traffic sent in both directions between hosts. Both applications communicate with one another to configure policies and relay classification results.

The system architecture for nmeta2 is illustrated in Figure 2. The traffic classification applications run on the Ryu controller. It manages OpenFlow v1.3 compliant switches that have connected to the controller by configuring their respective flow tables. This is used to forward packets through a network, be it to another switch, a host, the controller or to a DPAE. A single nmeta2 controller manages one or more DPAE.

The DPAE offers functionality whereby custom classifiers can be implemented in Python. Once a class is implemented, the DPAE instantiates the classifier when the application is loaded. As mentioned earlier in the paper, we chose five simple classifiers and use the feature set shown in Table I.

The controller instructs the DPAE to operate in one of two modes. While in active mode, all data plane traffic that arrives at a switch is forwarded to a DPAE instead of being forwarded to its destination. Once a DPAE has processed a packet, it gets

TABLE II
REFERENCE STANDARD FOR FIVE NETWORK TRAFFIC CLASSIFIERS BASED ON CAIDA AND ISCX DATASETS.

| Method | Accuracy | Precision | f-measure | Mean DR | Mean FPR |
|---|---|---|---|---|---|
| Random Forest | 0.997951721 | 0.962004963 | 0.954795172 | 0.990108069 | 0.029333478 |
| kNN | 0.998385248 | 0.985354048 | 0.948385248 | 0.990000645 | 0.027850928 |
| QDA | 0.997696021 | 0.981804961 | 0.915662191 | 0.989906291 | 0.014991284 |
| Naive Bayes | 0.997517048 | 0.971473281 | 0.878745778 | 0.989756706 | 0.024335124 |
| SVM | 0.998381771 | 0.895336052 | 0.830928575 | 0.989368817 | 0.012388389 |

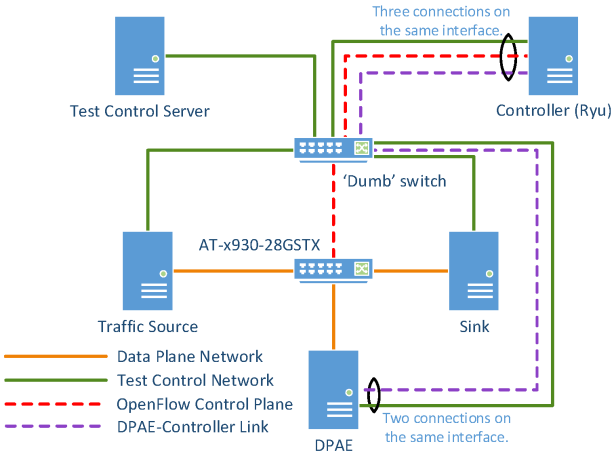


Fig. 1. Physical network testbed topology used for the on-line experiments.

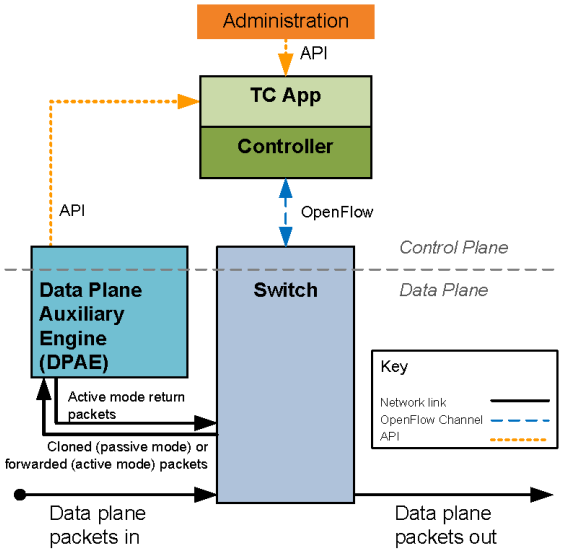


Fig. 2. System architecture of the *nmeta2* traffic classifier (TC). Diagram sourced from [18].

sent back to the switch where it is then forwarded towards its destination.

In contrast, passive mode informs the DPAE not to send packets back to the switch once they have been processed. This occurs because the packet is cloned after it arrives at the switch. The switch forwards one copy to its destination and the other to the DPAE.

The statistical classifiers begin their initialisation processes when they are instantiated. The classifiers undergo a training or learning phase before classifications can be made. The initialisation process not only includes the training phase of a classifier, but also the necessary steps to load and prepare training data before training may begin.

The statistical classifiers predict the class of a bidirectional flow each time a matching packet is encountered. The features used by the classifiers have a notion of a source and destination which roughly map to the notions of client and server that are used by the DPAE. These notions typically represent who initiated a session, namely the source or client. Given the DDoS attack context of the dataset being used, one cannot assume that an attack is commenced by the party who initiated the connection. Therefore when a classifier makes a prediction, the total number of bytes sent in the current direction will be used as the source bytes. This is done regardless of the label assigned to the direction of a flow.

Consider hosts A and B, a flow that was initialised by host A to host B and the selected kNN classifier as an example. At some point in time, host B sends a packet to host A. For the flow to be classified by the kNN classifier data referring to the `totalSourceBytes`, `totalDestinationBytes` and `flowDuration` is needed. As host A initiated the flow, one may be tempted to use the host A's byte count for the `totalSourceBytes` feature. Given the aforementioned assumption that an attack is started by the party initiating the connection, however, this may not result in an attack being detected. Therefore we use host B's byte count for the `totalSourceBytes` feature and host A's byte count for the `totalDestinationBytes` feature.

## IV. EVALUATION

An ideal DDoS detection requires classifiers that yield high accuracy, precision and DR in addition to having a low FPR. We evaluated the five chosen classifiers (Random Forest, QDA, Naive Bayes, kNN and SVM) in a live network setting and compared to the reference standard. The summary of the comparison is provided in Table II. Each classifier is tested under the following three validation scenarios: (i)10-fold cross validation repeated 10 times , (ii) 5-fold cross validation repeated 10 times and (iii) 5-fold cross validation with no repeats, the results in the next section show the average of the three validation scenarios.

### A. Results

The prediction performance measures used for the second classifier selection experiment were also used for online experiments. The DR and FPR were calculated for each classifier

alongside the f-measure to rank the classifiers. Tables III and IV present the results for each classifier with passive and active modes respectively.

*1) Accuracy & Precision:* Results from the live network exhibit a significant difference from the standard reference. In Table III, we see that the mean accuracy in passive mode differs from the reference standard between 6.9% – 11.2%, while the accuracy for active mode is closer to the reference standard 6.2% – 8.3%. The accuracy order in the live network is identical to the reference standard suggesting that the cause of the lower accuracy appears to be independent of the classifier used suggesting a systematic error as a possible explanation.

The accuracy of each classifier was roughly between 91% and 93% which may indicate very promising results. However, such results should be interpreted with care. Equation (3) showed how to calculate accuracy using the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). It is obvious that the calculated value is governed by the $TP$ and $TN$ terms in the numerator. This can be used to explain the high accuracy of the classifiers despite their low f-measure.

The high accuracy was attributed to the low mean FPR. The low mean DR for each classifier informs us that the number of TP predictions must have been smaller than the number of FN predictions. Similarly, the low mean FPR informs us that the number of TN predictions must have been larger than the number of FP predictions. Using these two pieces of information, one can infer that the high accuracy was due to a large number of TN predictions. This tells us that the classifiers were successful in correctly identifying non-malicious flows of traffic.

The precisions of the different classifiers in the live network were in line with the reference standard. Again, we see that the active mode achieves precision closer to the reference standard whereby the recorded average precision is around 22.9% – 24.8% lower compared to the reference standard. For passive mode, the precision results are worse, specifically 25.3% – 30.6% lower than the reference standard.

*2) f-measure:* The f-measure metric is frequently used to compare different classifiers for selecting the best classifier. The first observation from the live network results shows that the f-measure ranking of classifiers does not follow the standard reference in both passive (Table III) and active (Table IV) classification mode. Additionally, the difference in the orders of magnitude between the reference standard and live network results is larger than one might expect.

For the reference standard, the order (from highest to lowest f-measure) is

*Random Forest* $>$ *kNN* $>$ *QDA* $>$ *Naive Bayes* $>$ *SVM*.

For the live network with passive mode, the order is

*SVM* $>$ *Naive Bayes* $>$ *Random Forest* $>$ *kNN* $>$ *QDA,*

while for the live network with active mode the order is

*SVM* $>$ *Naive Bayes* $>$ *kNN* $>$ *Random Forest* $>$ *QDA*.

This discrepancy makes it difficult to make conclusive statements on which classifier works best for detecting DDoS and highlights one of the dilemmas faced by networking researchers, specifically the performance of a classifier is dependent on many network parameters on top of tuning algorithmic parameters (hyper-parameter optimisation).

*3) mean DR & mean FPR:* The mean DR for all classifiers was also very low. The highest mean DR was given by the SVM classifier when the DPAE was run in active mode. This value suggests that it could accurately identify roughly $15\%$ of the malicious flows (active mode using SVM).

The mean FPR results were more promising as lower values indicate better performance in this context. Furthermore, the lowest mean FPR was given by the kNN classifier when the DPAE was run in active mode. The disadvantage of using the DPAE in passive mode with packet sampling is that predictions are made with less information. Active mode puts the DPAE into a position where it has a closer relationship with the flows of traffic. Thus, the DPAE is closer to having perfect information, similar to the off-line experiments used for classifier selection.

## V. Discussion

The five statistical classifiers produced results that did not align with the initial expectations established via the reference standard. The low mean f-measure across all classifier-DPAE scenarios makes it difficult to determine what classifier performed the best. Despite this, the classifier-DPAE scenario with the highest f-measure was the SVM classifier with the DPAE configured in active mode.

The low mean f-measure statistics are matched with equally low mean DRs. However, it is important to consider each classifier's mean FPR as well, which got no larger than roughly 0.27%. The accuracy of each classifier fell within the range of 92-93%. While being satisfactory results, it is important to not let this paint a false picture of the predictive performance of each classifier.

Overall, the results also suggest that having the DPAE in active mode will produce results closer to the reference standard. Of the six feature sets (classifier-DPAE scenarios), two of the active-based classifiers were in the top three when ranking the classifiers by mean f-measure.

Considering the obtained results, the following observations can be made:

**Packet loss that may occur along the network should be taken into account as it affects the classifiers' performance significantly:** One observation in the live network evaluation is that the number of packets received by the sink is significantly fewer than the number of packets in the dataset's PCAP file. The highest mean number of packets received (recorded by the sink) was 24,326,964 packets (averages to around 70% for both datasets ), which is around 10 million packets less than the 34,983,042 packets of the PCAP file. Fewer classifications were expected to be made as not all packets were going to be subjected to statistical classification. Regardless, the lost packets offers a strong explanation for the poor prediction performance.

TABLE III
COMPARISON OF RESULTS REFERENCE STANDARD (REF) VS. LIVE NETWORK (LIVE). CLASSIFIER IN PASSIVE MODE.

| Classifier | Accuracy | | Precision | | f-measure | | Mean DR | | Mean FPR | |
|---|---|---|---|---|---|---|---|---|---|---|
| | REF | LIVE | REF | LIVE | REF | LIVE | REF | LIVE | REF | LIVE |
| Random Forest | 0.997951 | 0.893623 | 0.962004 | 0.720597 | 0.954795 | 0.000286 | 0.990108 | 0.015594 | 0.029333 | 0.002040 |
| kNN | 0.998385 | 0.915341 | 0.985354 | 0.752750 | 0.948385 | 0.000167 | 0.990000 | 0.027053 | 0.027850 | 0.000063 |
| QDA | 0.997021 | 0.915191 | 0.981061 | 0.741216 | 0.915191 | 0.000147 | 0.989906 | 0.024525 | 0.014284 | 0.000051 |
| Naive Bayes | 0.997048 | 0.906295 | 0.971281 | 0.733178 | 0.878778 | 0.001726 | 0.989706 | 0.026271 | 0.024324 | 0.000065 |
| SVM | 0.998771 | 0.921759 | 0.895052 | 0.622737 | 0.830575 | 0.011538 | 0.989817 | 0.117752 | 0.012388 | 0.002534 |

TABLE IV
COMPARISON OF RESULTS REFERENCE STANDARD (REF) VS. LIVE NETWORK (LIVE). CLASSIFIER IN ACTIVE MODE.

| Classifier | Accuracy | | Precision | | f-measure | | Mean DR | | Mean FPR | |
|---|---|---|---|---|---|---|---|---|---|---|
| | REF | LIVE | REF | LIVE | REF | LIVE | REF | LIVE | REF | LIVE |
| Random Forest | 0.997951 | 0.914583 | 0.962004 | 0.799841 | 0.954795 | 0.000594 | 0.990108 | 0.039379 | 0.029333 | 0.002041 |
| kNN | 0.998385 | 0.925757 | 0.985354 | 0.815341 | 0.948385 | 0.0003047 | 0.990000 | 0.0536431 | 0.027850 | 0.000092 |
| QDA | 0.997021 | 0.922185 | 0.981061 | 0.812772 | 0.915191 | 0.000293 | 0.989906 | 0.049846 | 0.014284 | 0.000078 |
| Naive Bayes | 0.997048 | 0.920368 | 0.971281 | 0.792402 | 0.878778 | 0.002599 | 0.989706 | 0.050377 | 0.024324 | 0.000085 |
| SVM | 0.998771 | 0.934685 | 0.895052 | 0.773007 | 0.830575 | 0.021384 | 0.989817 | 0.149199 | 0.012388 | 0.002501 |

**Delay, jitter and out-of-order packet arrivals directly affect the classifiers' performance:** Another observation from our deployment is that the classifier's performance is also dependent on the packet delays and jitter. Because each classification is conducted periodically and there are timers to expire a classification operation, some packets may arrive out of order thus contributing to either (i) misclassification or (ii) no classification, while both will eventually drive down the performance of the traffic classifiers. Our results showed that 2.8% of packets arrived out of order and the standard variation for jitter was 93.6ms, and perhaps, this helps explaining the poor classification results in the live network compared to the reference standard.

## VI. CONCLUSION

This paper evaluated five statistical classifiers on a physical network testbed. These classifiers did not classify traffic as accurately as when they were tested using offline datasets. It is important to note that the poor classification of traffic concerns the classifiers' abilities to detect malicious flows within the dataset. Each classifier was successful in identifying non-malicious flows. Looking beyond the performance of the classifiers, it appears that more attention needs to be paid to the networking environments where ML is deployed because not enough thought is given to how these ML algorithms are affected by the network environment they operate in.

## REFERENCES


[1] Check Point Research, "Check Point 2018 Security Report," Check Point Software Technologies Ltd., Tech. Rep., 04 2018. [Online]. Available: https://blog.checkpoint.com/2018/04/16/2018-security-report-97-companies-unprepared-cyber-attacks/

[2] Cisco and/or its affiliates, "Cisco 2018 Annual Cybersecurity Report," Cisco Systems, Inc., Tech. Rep. C11-739110-00, 08 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf

[3] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli, "A statistical approach to IP-level classification of network traffic," in *Proceedings of the IEEE International Conference on Communications (ICC)*, vol. 1, Istanbul, Turkey, 11-15 June 2006, pp. 170–176.

[4] B. Ng, M. Hayes, and W. K. G. Seah, "Developing a Traffic Classification Platform for Enterprise Networks with SDN: Experiences & Lessons Learned." in *Proceedings of the IFIP Networking Conference (IFIP Networking)*, Toulouse, France, 20-22 May 2015, pp. 1–9.

[5] M. Stevens, B. Ng, D. Streader, and I. Welch, "Global and local knowledge in SDN," in *Telecommunication Networks and Applications Conference (ITNAC), 2015 International*. IEEE, 2015, pp. 237–243.

[6] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.

[7] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton, "Seeing Through Network-Protocol Obfuscation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, Colorado, USA, 12-16 October 2015, pp. 57–69.

[8] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 4 2008.

[9] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *Proceedings of the 2008 ACM CoNEXT Conference*, Madrid, Spain, 9-12 December 2008, pp. 11:1–11:12.

[10] N. Williams, S. Zander, and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, Oct. 2006.

[11] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian Neural Networks for Internet Traffic Classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 223–239, Jan 2007.

[12] T. Thapngam, S. Yu, and W. Zhou, "DDoS discrimination by Linear Discriminant Analysis (LDA)," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, Maui, Hawaii, 30 January - 2 February 2012, pp. 532–536.

[13] Michie, D. and Spiegelhalter, D.J. and Taylor, C.C., *Machine Learning, Neural and Statistical Classification*, Michie, D. and Spiegelhalter, D.J. and Taylor, C.C., Ed. Ellis Horwood, 1994.

[14] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, New York, 2008.

[15] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "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 (IMC)*, Taormina, Sicily, Italy, 25-27 October 2004, pp. 135–148.

[16] scikit-learn developers. (2014) 1.6. Nearest Neighbors. http://scikit-learn.org/stable/modules/neighbors.html (Accessed on 09/08/2016).

[17] M. Bramer, *Principles of Data Mining*, 2nd ed., ser. Undergraduate Topics in Computer Science. Springer London, 2013.

[18] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable Architecture for SDN Traffic Classification," *IEEE Systems Journal*, 2017.