CBR - Boosting Adaptive Classification By Retrieval of Encrypted Network Traffic with Out-of-distribution

Amir Lukach, Ran Dubin, Amit Dvir
Department of Computer Science
Ariel Cyber Innovation Center
Ariel University, Israel
amirluckach@gmail.com, rand, amitdv@ariel.ac.il

Chen Hajaj
Department of Industrial Engineering and Management
Data Science and Artificial Intelligence Research Center
Ariel University, Israel
chenha@ariel.ac.il

***Abstract*—Encrypted network traffic Classification tackles the problem from different approaches and with different goals. One of the common approaches is using Machine learning or Deep Learning-based solutions on a fixed number of classes, leading to misclassification when an unknown class is given as input. One of the solutions for handling unknown classes is to retrain the model, however, retraining models every time they become obsolete is both resource and time-consuming. Therefore, there is a growing need to allow classification models to detect and adapt to new classes dynamically, without retraining, but instead able to detect new classes using few shots learning [1]. In this paper, we introduce Adaptive Classification By Retrieval *CBR*, a novel approach for encrypted network traffic classification. Our new approach is based on an ANN-based method, which allows us to effectively identify new and existing classes without retraining the model. The novel approach is simple, yet effective and achieved similar results to RF with up to 5% difference (usually less than that) in the classification tasks while having a slight decrease in the case of new samples (from new classes) without retraining. To summarize, the new method is a real-time classification, which can classify new classes without retraining. Furthermore, our solution can be used as a complementary solution alongside RF or any other machine/deep learning classification method, as an aggregated solution.**

***Index Terms*—Malware detection, Approximate Nearest Neighbors, Encrypted Network Traffic Classification, Out-of-distribution, Classification By Retrieval, Few shots learning**

I. Introduction

Classical Machine Learning (ML) and Deep Learning (DL) models are applicable in the scope of encrypted network traffic classification [2]–[9], and thus served as the baseline for classifying encrypted traffic. Due to recent massive changes in the internet protocols, e.g., QUIC [10], HTTP/3, TLS 1.3 and DoH [11], Deep Packet Inspection (DPI) traditional classification methods, which leverage DNS and Service Name Indicator (SNI), to identify encrypted network traffic, will soon no longer be usable. Therefore, advanced encrypted traffic flow classification algorithms are needed [12]. Recently, the classification models adopted various data representations, including a growing number of works that have used Natural Language Processing (NLP) [2] techniques, such as transforming the flow into a language to use word embedding [3]–[6], while others have converted the network flow into an image to harness image processing techniques and equivalent DL architectures [7]–[9], [13].

In order to validate that an encrypted network traffic classification model will efficiently classify unknown samples correctly, there are a few requirements [14]: **I.** Acquire a large amount of training data for the new class, **II.** Add it to the dataset, which was initially used to train the classifier, and **III.** Retrain the classifier on the combined data set. Regarding ML (and especially DL) models, every time they become obsolete is both resource and time-consuming, especially when the application is complex. Therefore, there is a growing need to allow classification models to detect and adapt to new classes dynamically, without retraining [15]. One way of allowing the above is using Nearest neighbor search, one of the most well-known tools in many research areas [16], [17]. In some cases, a generic nearest neighbor search under a suitable distance or measure of similarity offers dramatic quality improvements [18]. K-Nearest Neighbors algorithm (KNN), by a plurality of its neighbors, is where the output is the class's label, and k is a positive (usually small) integer. To obtain efficient algorithms, one may use ANN [19] in which the returned neighbors may be an approximation of the true nearest neighbors. Usually, this means that the answer to finding the nearest neighbors to a query point is judged by the distance of the query point to the set of its true nearest neighbors. ANN [20] is a variation of KNN that aims to limit the number of training samples that each new test point is compared with before returning a result.

In this paper, we present a vector-based data representation that classifies encrypted network traffic and can identify and add new classes without the need for retraining. Our main motivtion was to find a balanced solution, the ability to easily learn new classes, that provides feasible and accurate results for different types of encrypted network traffic classification challenges. To achieve this goal, our approach relies on the ANN method, which excels in quickly matching a new vector against a collection of already labeled and indexed vectors. This matching process is based on the principle of Euclidean similarity distance (there are other distance metrics, such as Cosine distance) [21]. First, it performs a fast real-time classification; second, it allows users and security software to accurately identify new classes without the need for retraining the mode. Our novel approach is based on the idea that if a sample distance to other samples is higher than a defined threshold, we classify it as a new class. By that, we omit the necessity to retrain the model if new classes or instances are introduced, as the vector search process is dynamic. This way, our system can extend itself using only a few samples from a new class. Note that our proposed solution uses only statistical features, so it should be robust to future planned protocol changes. To summarize, the contributions of this paper are as follows:

1) To the best of our knowledge, this is the first paper that suggests encrypted network classification based on ANN that handles few shots learning and OOD.
2) Identification of network anomalies for information security purposes, by classifying them as new classes.
3) Fast detection and addition of new classes without retraining using a real-time classification model that needs only a few shots from the new classes, while still providing accuracy that is comparable to standard ML models (e.g., RF). Note, that when combining OOD to our solution, in some cases it even outperformed RF. In addition, it can also be used with RF as an aggregated solution.
4) Preparing a classification solution that should comply with fully encrypted networks.

II. Related work

In recent years, DL models have become the prominent method for network traffic classification [22]–[25]. The DL's work has multiple scopes and domains such as classification [26]–[29]. Some works have converted the network flow into an image to harness image processing techniques and equivalent DL architectures [7], [8], [13], [27], [30], [31], while others used NLP [6] or graph neural networks [32]. In the cyber domain, works tackle the task of malware network traffic detection (benign, malicious) and classification [8], [24], [33]–[36]. In addition, some previous works suggest using classification by retrieval to solve specific video traffic analysis challenges, like motion detection, Chan et al. [37], and audio classification and segmentation, which allows real-time audio classification into basic types, Zhang et al. [38], however, these are very domain-specific solutions. Although DL's architectures started to replace ML for traffic classification, in [39] the authors presented a comparative experiment between ML/DL algorithms that shows, in some cases, that ML algorithms such as RF [40] are more than enough. All the above works tackle the classification problem without discussing the challenge of retraining the model, which is needed due to a changing network traffic landscape. Retraining a machine learning model involves updating the model to accommodate the new knowledge, which is necessary to perform well, most of the works use datasets with only a few classes, e.g. [7], [41], they use per-flow features and do not consider scenarios where new applications are progressively added to models. Therefore, the focus of these works is only on the problem of creating the most accurate classifier given immutable data for both the number of classes and the data for each class. These systems are based on creating a new training set and training a new model from scratch. It may, however, be inefficient and require high computation performance, to update them with new classes' classification. The K-Nearest Neighbors (KNN) is one kind of lazy classification algorithm without the process of classifier training. By learning-to-hash algorithms, the KNN-based classification can be mapped to the hash table searching whose execution time and memory cost are both acceptable. Qi et al. [42] presented lightweight IoT traffic classification based on KNN [43]. Ma et al. [44] proposed a method, based on the K-nearest neighbor (KNN) algorithm, which only needs a small amount of data to train a model. Moreover, the authors also presented a three-layer classification framework for encrypted network flows. ANN [20] is a variation of KNN that aims to limit the training sample number that each new test point is compared with, before returning a result. Many efficient ANN implementations have been developed with diverse approaches, such as dimension reduction [45], locality-sensitive hashing [46], and compressed sensing [19], but none of them use the advantages of the ANN in the field of encrypted network classification.

III. Methodology

In this section, we will explain how our solution (CBR) tackles the problem of retraining, while still providing a fast and accurate classification. Our goal is to create an adaptive, encrypted traffic classifier, which will be able to accurately identify existing, and new classes in a fast, real-time, and dynamic method. To achieve fast classification, we have chosen ANN, and have used a distributed search engine for storing and retrieving our samples, since it supports real-time search performance [47] (note that our solution also supports using any other ANN search algorithm). Moreover, our solution has the following benefits: near real-time operations, such as reading or writing data usually take less than a second to complete, and high performance. Using the distributed nature of the search engine enables it to process

Feature	Slots in vector	Description	Used In
Bits per peak	3	Summary of bits of every ”peak” in data coming from destination to source	
First packets sizes	30	First 30 packet sizes with sign (+ or -) by direction of communication	
Beaconing	20	Sum of packet size where the source is more active than the destination in 5 seconds windows	BOA & MTA
Bandwidth	20	Minimum and maximum delta of TCP window size in 5 seconds windows	
Statistics of packet sizes	4	Min, max, mean, STD of packet sizesa	BOA & MTA
Size’s delta between packets	2	Mean and STD deltas	BOA & MTA
Packets per second	2	Packets per second forward and backward	
Inter-arrival time	9	Min, max, and mean of bidirectional, forward, and backward	
Silence windows	1	Amount of silence windows longer than 1 second or 10 ms every 1 or 0.5 seconds	
Amount of ACK packets	1	Bidirectional count of TCP packets which contain ACK flag	
Big requests	1	Amount of client-to-server messages bigger than 200-byte ending session with a new request	
Wavelet	90	FFT coefficient on first 90 packet sizes received	BOA & MTA

TABLE I: Feature sets.This table describes the features vector we used, the size in the vector of each of the features, and for which datasets we used it.

large volumes of data in parallel, and find the best matches for our queries [47].

The CBR’s architecture steps are as follows (as can be seen in Figure 1): The first step is to read the network data, afterwards we organize the data by connections 5-tuple (source IP, destination IP, protocol, source port, and destination port), from each sample, we extract the required features (see Table I) and normalize them, perform feature selection to use the minimal amount of features, to provide the fastest solution, use Random Forest as one of the inputs for classifying the classes, perform semi-supervised learning to find OOD and use it as input for CBR, use CBR to classify new, and existing classes and OOD. Then, inputs from RF and CBR are combined in order to classify current classes, new classes, and OOD (for new classes and OOD we use input from CBR, and use RF for the rest of the classification). In the training phase, to build our vector search database, we split the data into a training set (70%) and a test set (30%), and add the training set features’ to the Elastic search database [47]. For each sample from the test set, the model predicts the label. The inference phase of CBR is done by ANN queries from the database. Each ANN query selects K’s closest samples, and we choose the class with the most hits.

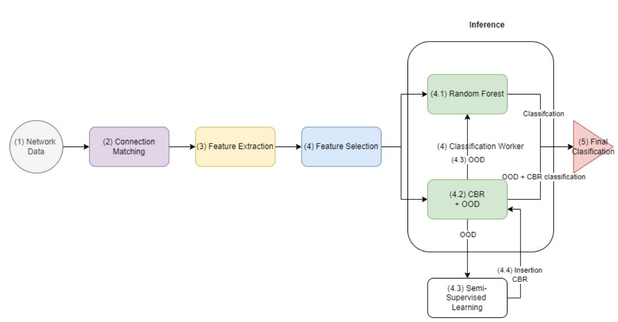


Fig. 1: CBR Architecture

To summarize, in the training phase, each sample is transformed into a vector with a label. The model saves each training sample vector index and uses the label to map it to a class. The inference phase of CBR is responsible for labeling each test sample with the matching class by using the distance from the test sample to get k samples from the database and then using the label of the sample with the max hits to decide on the label. Furthermore, if the distance between the current tested vector features to the closest class is larger than a defined threshold, we add a new class. In our case, we use the Euclidean distance metric to query the elastic search for K closest samples and select the class with the maximum hits’ score. Furthermore, to prevent adding new classes from Out Of Distribution samples, we have used semi-supervised learning (we manually labeled some of the OOD) for Out-Of-Distribution detection, for samples, which are very distant from their closest class and have used existing dataset’s features, without calculating features from related packet capture files. ANN methods have been used in the past for OOD detection, e.g. Sun et al. [48], but in our case, we compared RF, VS ANN with OOD which was used to improve the classification of known classes and not just to detect the OOD. If we look at ANN without OOD, then, in this case, every test sample is labeled with the closest class, no matter how distant they are from the closest class.

The latter is displayed in Figure 2. Moreover, there is also an option to use our solution, alongside RF as part of the same prediction flow, in which our solution can be used to remove OOD and combine with the results we receive from RF, as depicted in Figure 1.

IV. Experimental Design

We conducted a set of experiments to evaluate the effectiveness of the ANN-based approach for a set of classification tasks on encrypted network traffic from multiple known public datasets. Specifically, we looked at both malicious and benign datasets. Using the malicious datasets, we classified the malware family. For the benign datasets, we evaluated the Operation System (OS), browser classification, and the application’s classification. The goal of our evaluation was, first to assess the slight decrease in our classifier’s performance due to learning new classes, while

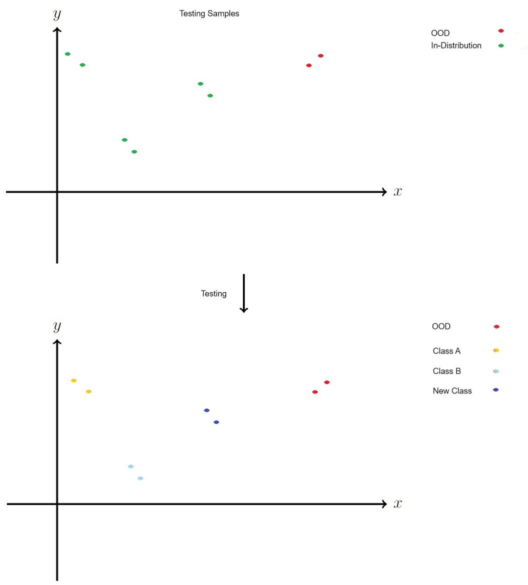


Fig. 2: ANN with OOD

classifying existing classes when comparing them to a classical machine learning classifier (e.g., RF). Second, to show the robustness of our ANN-based classifier in the classification of new classes.

A. Datasets

We used two common datasets for our evaluations: BOA [26], and MTA [49]. The **BOA** dataset was presented in [26] where the authors collected the data for more than two months in their lab, using a selenium web crawler for browser traffic. The dataset contains applications, such as YouTube and Facebook, labeled as browser traffic, and Dropbox and TeamViewer, labeled as non-browser traffic. The dataset contains more than 20,000 sessions. This dataset contains information on browsers, OSs, and applications with and without their correlated browser. The average duration of a session was 518 seconds, and on average, each session had 520 forward packets (the average forward traffic size was 261K bytes) and 637 backward packets (the average backward traffic size was 615K bytes). Almost all of the flows are TLS encrypted. Examples of works that used this dataset include [26], [50]. The **MTA** dataset is a website (blog) [49] that includes many types of malware infection traffic for analysis. The website contains many types of malware, such as ransomware and exploit kits. As of 2013 to date, the blog is updated daily with relevant malware traffic, continuously adding more samples to the dataset. Using Intrusion-Detection Systems (IDS) and Antivirus software, every binary file in the PCAPs has been confirmed as malicious. Papers such as [24], [33], [51] have used this dataset for malware detection.

V. RESULTS

In this section, we present our experimental results, which include multiple experiments on BOA and MTA datasets. For each dataset, we first find the optimal number of packets from which results do not significantly improve, then calculate the minimal features' set by performing a feature selection [52] (for more details on the process of the feature selection please visit Github project page [53]. Moreover, we present the results of our model with new samples, and the influence of the searching algorithm on the results. Note that we have chosen statistical features because in the near future, the traffic will not include clear indicators about the service name, so we have to identify it using its statistical footprint.

A. CBR Vs RF - BOA dataset

In the first experiment, we wanted to check the influence of the number of packets on our solution compared to RF. Therefore, we used the entire feature sets (see Table I) and increased the number of packets from each flow until the accuracy of our solution stopped improving. Figure 3 depicts the accuracy results of our approach as a function of the number of packets. These results motivated us to use only 10 packets from each flow.

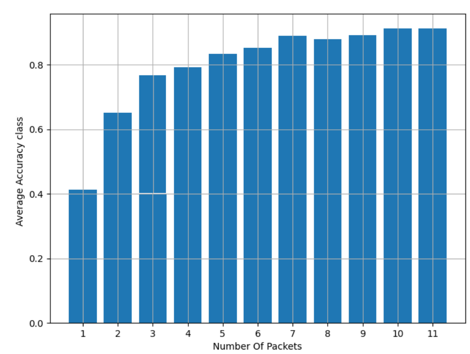


Fig. 3: CBR accuracy results as a function of the number of packets - BOA dataset. As can be seen, the accuracy stops improving after 10 packets.

After setting the number of packets, we wanted to compare our approach to RF, as can be seen in Table II. The Table presents the results of our ANN model for the BOA dataset with the optimal number of processed packets (10 packets) vs RF.

Note that we do not aim to beat RF, but rather perform as close as possible to RF while our solution does not need retraining once a new class appears. Our results confirm this by showing that our CBR classification results are very close to the results that RF provides, which was one of our goals in this experiment.

BOA Classes	Our Approach			RF		
	Prec	Rec	F1	Prec	Rec	F1
win	0.96	0.95	0.95	1	0.99	0.99
OSX	0.92	0.97	0.94	0.92	0.99	0.95
Linux	0.98	0.98	0.98	1	0.98	0.99
CR	0.91	0.90	0.91	0.96	0.98	0.97
FF	0.91	0.91	0.91	1	0.99	0.99
IE	0.97	0.97	0.97	0.97	0.98	0.97
Safari	0.95	0.97	0.96	1	1	1

TABLE II: CBR model for BOA dataset using 10 packets. The results are very close to RF's results.

Feature	Description
dst2src mean piat	Minimal arrival time between packets from the dst to src
ps 9	Packet size of the 9th packet
ps 21	Packet size of the 21th packet
ps 24	Packet size of the 24th packet
ps 28	Packet's size of the 28th packet
beacon 10	The 10th index of max packet size (dst & src)
beacon 11	The 11th index of max packet size (dst & src)
beacon 15	The 15th index of max packet size (dst & src)
wavelet 0	Using the first coefficient on FFT of the packet sizes
wavelet 16	Using the 16th coefficient on FFT of the packet sizes
wavelet 7	Using the 7th coefficient on FFT of the packet sizes

TABLE III: Feature selection for BOA. These features are the minimal features' set used for BOA dataset classification for CBR classification.

	Pr	Rec.	F1-SC	RF Pr	RF Rec.	RF F1-SC
Win	0.96	0.95	0.95	0.99	0.99	0.99
OSX	0.86	0.9	0.92	0.92	0.98	0.95
Linux	0.98	0.98	0.98	0.99	0.98	0.99
CR	0.9	0.9	0.9	0.96	0.97	0.96
FF	0.91	0.91	0.91	0.99	0.98	0.99
IE	0.97	0.98	0.97	0.97	0.98	0.97
Safari	0.95	0.95	0.95	0.99	0.99	0.99

TABLE IV: CBR model for BOA dataset using 10 packets with minimal features. The results are very close to RF's results.

	Pr	Rec.	F1-SC	RF Pr	RF Rec.	RF F1-SC
Infostealer	0.95	0.96	0.95	0.96	0.97	0.96
Dropper	0.93	0.92	0.92	0.95	0.95	0.95
ACC		0.96			0.97	
Dridex	0.84	0.79	0.81	0.86	0.85	0.55
Emotet	0.92	0.93	0.92	0.93	0.94	0.93
Hancitor	0.96	0.96	0.96	0.96	0.96	0.96
Icedid	0.85	0.93	0.84	0.87	0.92	0.87
Qakbot	0.92	0.94	0.93	0.92	0.94	0.93
Valak	0.92	0.94	0.93	0.92	0.94	0.94
zloader	0.86	0.88	0.87	0.87	0.88	0.87
ACC		0.93			0.95	

TABLE V: CBR model for MTA dataset using 98 packets

	Pr	Rec.	F1-SC	RF Pr	RF Rec.	RF F1-SC
Infostealer	0.95	0.96	0.95	0.95	0.97	0.95
Dropper	0.93	0.92	0.92	0.94	0.94	0.94
Accuracy	0.96					
Dridex	0.84	0.79	0.81	0.85	0.82	0.82
Emotet	0.92	0.93	0.93	0.93	0.93	0.93
Hancitor	0.95	0.95	0.95	0.95	0.95	0.95
Icedid	0.85	0.92	0.84	0.86	0.92	0.86
Qakbot	0.92	0.93	0.93	0.93	0.93	0.93
Valak	0.92	0.93	0.93	0.92	0.93	0.93
zloader	0.86	0.88	0.87	0.87	0.88	0.87
Accuracy	0.93					

TABLE VI: CBR model minimal features set for the MTA dataset using 98 packets

	F1-Score (before adding new class)	F1-Score
Beacon		0.95
Infostealer	0.97	0.96
Dropper	0.92	0.91
Cobalt Strike		0.95
Dridex	0.81	0.79
Emotet	0.93	0.93
Hancitor	0.97	0.97
Icedid	0.84	0.82
Qakbot	0.93	0.93
Valak	0.93	0.93
zloader	0.87	0.84

TABLE VII: CBR model for MTA's results for Cobalt Strike. The results show, that accuracy of identifying current classes is not worsen, by the addition of the new class.

B. CBR Vs RF - Minimal Features set on BOA dataset

In the second evaluation, we did a supervised filter-based feature selection [52], to obtain the minimal number of features that provide maximal performance using the 10 packets from each sample. The results of both algorithms (ours and RF) with the minimal feature set used for both classifiers are depicted in Table IV, while the feature set is depicted in Table III, and as we can see that RF performs slightly better than our algorithm. From the above experiments, we see that based on the BOA dataset, our CBR model achieved results that are close to RF.

C. CBR with OOD Vs RF - BOA dataset

In this section, we have used a different strategy. We used labeled applications, and their features, and used all their features as is, instead of calculating features from packet capture files, while some of these applications are labeled as not categorized (e.g. Netflix traffic), as shown in Table 1. To complete the picture, note that each sample contains its features, and each one contains packet size, and flow direction (inbound/outbound). Afterward, we used CBR and labeled the samples whose distances from their class are larger than a threshold as OOD [54] as depicted in Figure 4 , and the results are depicted in Table VIII:

Class	Pr	Rec.	F1-SC	RF Pr	RF Rec.	RF F1-SC
Dropbox	0.9	1.0	0.95	0.91	0.91	0.91
Facebook	0.97	1.0	0.98	0.675	0.675	0.675
google	0.79	1.0	0.88	0.95	0.95	0.95
Microsoft	1.0	1.0	1.0	0.91	0.91	0.91
TeamViewer	0.99	1.0	1.0	0.95	0.95	0.95
Twitter	0.99	0.98	0.98	0.99	0.99	0.99
Youtube	0.67	0.88	0.76	0.91	0.91	0.91

TABLE VIII: BOA applications CBR classification. As seen in the table, in some cases CBR provides more accurate results than RF, and in some cases, it is vice versa.

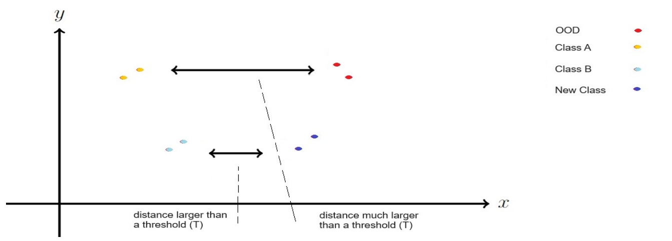


Fig. 4: CBR with OOD. Classify samples as new classes or OODs according to their distances from the nearest classes.

As we can see from the results, for some classes our algorithm outperforms RF, and for others it is vice versa.

D. CBR Vs RF - MTA dataset

In this section, similar to the first experiment, we wanted to check the influence of the number of packets on our solution. Therefore, we used the entire feature sets and increased the number of packets from each flow until the accuracy of our solution stopped improving (this happened after 98 packets), as depicted in Figure 5. We present the optimal classifier results, using 98 packets (on average around 50-60 KB) for MTA dataset in Table V. Again, we see that RF as an ensemble method slightly outperforms our CBR. Notice, that MTA consists of a main class, which includes InfoStealer and Dropper, and the rest are secondary classes.

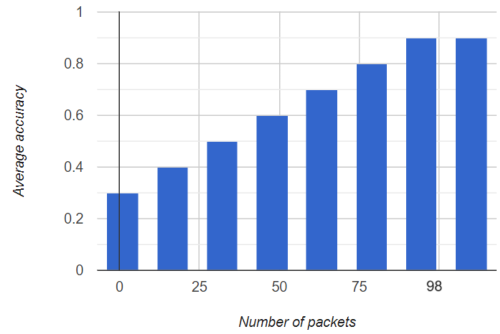


Fig. 5: CBR accuracy results as a function of the number of packets - MTA dataset. Notice, that after 98 packets, the accuracy does not improve.

E. CBR Vs RF - Minimal Features set on MTA dataset

In this section, we performed a supervised filter-based feature selection [52] on MTA dataset, to obtain the minimal number of features, which provide maximal performance using 98 packets. The results are depicted in Table VI (the list of features are the same features for RF and CBR, and are depicted in Table IX).

From the above experiments, we can see that also based on the MTA dataset, our CBR model achieved results that are close to RF. Furthermore, after presenting that our model results are close to RF, in the following section we will show the ability of our model to adapt to new classes without any retraining phase.

F. CBR New Classes Classification

So far we tested our approach on known classes. In the following section, we present the results of our approach in the case of new classes of malware (i.e. Cobalt Strike), first seen by the model. In this experiment, we did not mention RF results, since it is not able to deal with new classes dynamically, only predict based on trained classes. The result are shown in Table VII, from the results, we can see there is a slight decrease in the accuracy when testing samples from new classes, however, all the classes are classified accurately.

G. ANN Benchmark

In this section, we provide our final evaluation which is a comparison between the ANN search algorithms. We have used vector search, which provided the best results in most of the experiments we did, however, in some cases, the best results were produced by the Ball tree [55] algorithm. The commonly used ANN algorithms are shown in Figure 6. The list of search algorithms is as follows:

Feature	Description
window delta 11	In a 5 seconds window, the 11th index of the max difference between packet size sent/received in the same direction
wavelet 9	Using the 9th coefficient on FFT of the packet sizes in a 10 seconds window
ps 29	Packet size of the 29th packet our of first 30 packets in the same direction
ps 6	Packet's size of the 6th packet our of first 30 packets in the same direction
wavelet 12	Using the 12th coefficient on FFT of the packet sizes in a 10 seconds window
window delta 10	In a 5 seconds window, the 10th index of the max difference between packet's size sent/received in the same direction
wavelet 18	Using the 18th coefficient on FFT of the packet sizes in a 10 seconds window
bidirectional mean piat ms	average arrival time between packets in both directions
wavelet 5	Using the 5th coefficient on FFT of the packet sizes in a 10 seconds window
ps 4	Packet's size of the 4th packet our of first 30 packets in the same direction

TABLE IX: Feature selection for MTA. These features' are the minimal features set used for MTA dataset for CBR classification. Note, that the number near each features shows the granularity of its usage, for example, **ps 4**, means the 4th packet's size of the first 30 packets in the same direction.

1) Ball tree - partitions data points into a nested set of balls (a ball is a solid figure bounded by a sphere).
2) KD tree - a binary tree, in which each node represents a k's dimensions point [56].
3) annoy - uses file-based data structures mapped into the memory, to allow searching for points, which are close to a query point [57].
4) Bruteforce - blas - Uses brute force to calculate Euclidean distances and sorts the results [58].
5) cKDTree - improved kd tree construction algorithm [59].
6) faiss - AI similarity search [60].
7) hnswlib - fast approximate nearest neighbor search [61].
8) nearpy - a simple yet modular little framework for ANN search [62].
9) redis - uses vector similarity in a key/value in memory database [63].
10) rpforest - uses a forest of random projection trees [64].
11) elastic - incorporates KD trees to support searches on geospatial and numeric data.

From the results, we can see that the algorithm we have used gives the best results for the MTA dataset, or is close to the best possible results, in the case of the BOA dataset, in which Bruteforce - blas gave the best results. We chose elastic, since it not only usually provides the best results for the selected datasets, but also has a near real-time speed for processing samples.

VI. Discussions and Future Perspectives

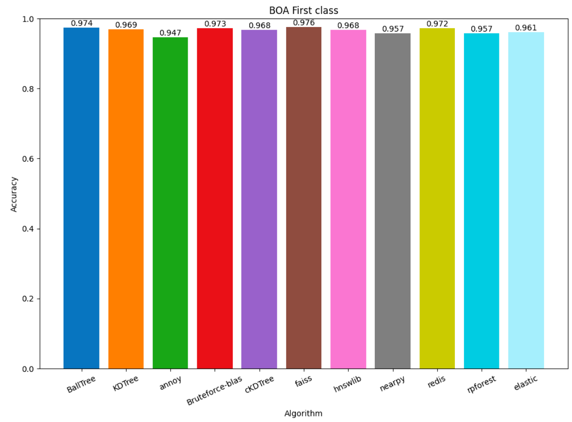
In this paper, we have shown how our approach, which relies on the ANN method, can achieve the following goals - detect malware activity on encrypted network traffic, classify the malware type and name, create a simple model that can be maintained easily, detect unknown traffic, classify it as new classes, creates an alternative solution for retraining. Our plans for future work include: features selection for behavioral features needed for malware classification, features selection for behavioral features needed for application classification, feature selection for behavioral features needed for application type classification (chat, VoIP [65], data download, VOD [66], etc.), understanding the impact of the order of the features, on the classification's results, optimize runtime performance, use *CBR*'s classification on additional datasets, benchmark our solution VS additional ML/DL methods, as we can see in Figure 6, compare to additional ANN search algorithms, our algorithm provided the best results most of the time. In addition, CBR also uses the elastic database, which provides very fast data access, and fast run-time performance. Furthermore, To the best of our knowledge, this is the first paper to show usage of few shots learning, and OOD, and this can be a basis for future research, in order to improve few shots learning accuracy.

VII. Acknowledgment

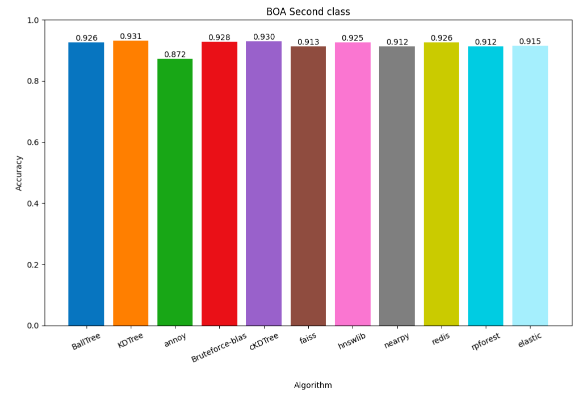
This work was supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office, the Israel Innovation Authority, and the ENTM consortium, and is patent-based on US Provisional Patent Application No. 63/448,460.

References

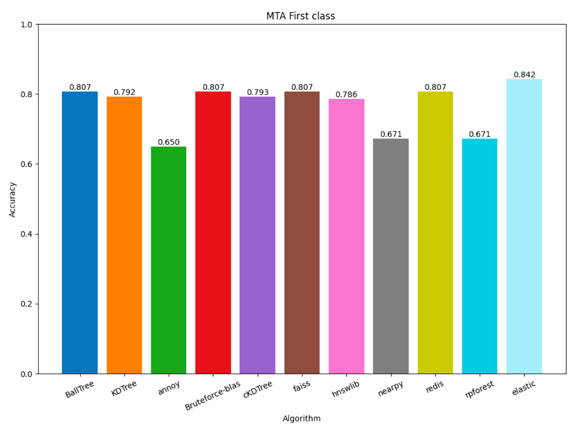
[1] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.
[2] Oracle, "Quic transport protocol rfc9000," 2023. [Online]. Available: https://www.oracle.com/il-en/artificial-intelligence/what-is-natural-language-processing/
[3] E. L. Goodman, C. Zimmerman, and C. Hudson, "Packet2vec: Utilizing word2vec for feature extraction in packet data," *arXiv preprint arXiv:2004.14477*, 2020.
[4] J. Li, H. Zhang, and Z. Wei, "The weighted word2vec paragraph vectors for anomaly detection over http traffic," *IEEE Access*, vol. 8, pp. 141 787–141 798, 2020.
[5] R. Corizzo, E. Zdravevski, M. Russell, A. Vagliano, and N. Japkowicz, "Feature extraction based on word embedding models for intrusion detection in network traffic," *Journal of Surveillance, Security and Safety*, vol. 1, no. 2, pp. 140–150, 2020.
[6] R. Bar and C. Hajaj, "Simcse for encrypted traffic detection and zero-day attack detection," *IEEE Access*, 2022.
[7] T. Shapira and Y. Shavitt, "Flowpic: A generic representation for encrypted traffic classification and applications identification," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1218–1232, 2021.



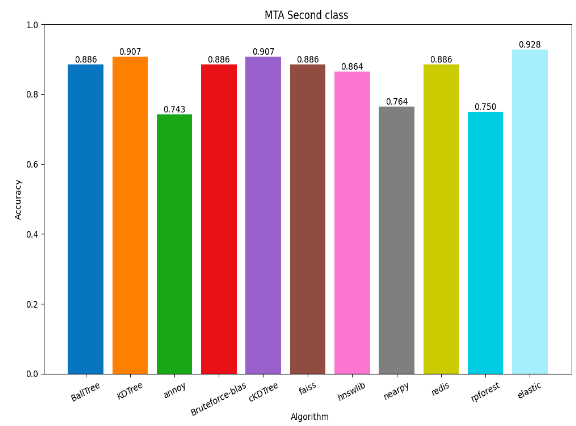
(a) BOA First Class's ANN Benchmark



(b) BOA Second Class's ANN Benchmark



(c) MTA First Class's ANN Benchmark



(d) MTA Second Class's ANN Benchmark

Fig. 6: ANN-Benchmarks. Notice, that for the MTA dataset, elastic, which is the search algorithm we use provides the best results. As for the BOA dataset, for which brute force approach provides the best results. This can be the case, in which classes are close to each other, so a brute force approach can work well.

[8] G. Marín, P. Casas, and G. Capdehourat, "Deepmal - deep learning models for malware traffic detection and classification," *CoRR*, vol. abs/2003.04079, 2020.

[9] E. Horowicz, T. Shapira, and Y. Shavitt, "A few shots traffic classification with mini-flowpic augmentations," in *IMC*, 2022, pp. 647–654.

[10] Google, "Quic, a multiplexed transport over udp," 2023. [Online]. Available: https://www.chromium.org/quic/

[11] T. Böttger, F. Cuadrado, G. Antichi, E. L. Fernandes, G. Tyson, I. Castro, and S. Uhlig, "An empirical study of the cost of dns-over-https," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 15–21.

[12] D. Serpanos and T. Wolf, "Transport layer systems," 2011. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/flow-classification

[13] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *ICOIN*, 2017, pp. 712–717.

[14] A. Moschitti. Updating neural networks to recognize new categories, with minimal retraining. [Online]. Available: https://www.amazon.science/blog/updating-neural-networks-to-recognize-new-categories-with-minimal-retraining

[15] J. Schulz, C. Veal, A. Buck, D. Anderson, J. Keller, M. Popescu, G. Scott, D. K. Ho, and T. Wilkin, "Extending deep learning to new classes without retraining," in *DSMEOBT XXV*, vol. 11418, 2020, pp. 13–26.

[16] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas, "Fast nearest neighbor search in medical image databases," 1998.

[17] B. Bustos, O. Deussen, S. Hiller, and D. Keim, "A graphics hardware accelerated algorithm for nearest neighbor search," in *ICCS*, 2006, pp. 196–199.

[18] L. Boytsov, D. Novak, Y. Malkov, and E. Nyberg, "Off the beaten path: Let's replace term-based retrieval with k-nn search," in *CIKM*, New York, NY, USA, 2016, p. 1099–1108.

[19] M. Aumüller, E. Bernhardsson, and A. Faithfull, "Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," *Information Systems*, vol. 87, p. 101374, 2020.

[20] Apache, "Approximate nearest neighbors algorithm," 2023.

[21] S. A. Gokte, "Most popular distance metrics used in knn and when to use them," 2023. [Online]. Available: https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

[22] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *ICOIN*, 2017, pp. 712–717.

[23] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *ISI, Beijing, China, July 22-24*, 2017, pp. 43–48.

[24] D. Kim, J. Han, J. Lee, H. Roh, and W. Lee, "Poster: Feasibility of malware traffic analysis through tls-encrypted flow visualization," in *ICNP 2020, Madrid, Spain, October 13-16*, 2020.

[25] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *INFOCOM Paris, France,*

April 29 - May 2, 2019, pp. 1171–1179.

[26] R. Dubin, A. Dvir, O. Pele, J. Muehlstein, Y. Zion, M. Bahumi, and I. Kirshenboim, "Analyzing https encrypted traffic to identify user's operating system, browser and application," in *CCNC*, Jun. 2017.

[27] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

[28] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE TIFS*, vol. 13, no. 1, pp. 63–78, Jan 2018.

[29] A. J. Pinheiro, J. de M. Bezerra, C. A. Burgardt, and D. R. Campelo, "Identifying iot devices and events based on packet length from encrypted traffic," *Computer Communications*, vol. 144, pp. 8–17, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419300052

[30] O. Salman, I. H. Elhajj, A. I. Kayssi, and A. Chehab, "Data representation for CNN based internet traffic classification: a comparative study," *Multim. Tools Appl.*, vol. 80, no. 11, pp. 16 951–16 977, 2021.

[31] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, 2019.

[32] B. Pang, Y. Fu, S. Ren, Y. Wang, Q. Liao, and Y. Jia, "Cgnn: Traffic classification with graph neural network," 2021. [Online]. Available: https://arxiv.org/abs/2110.09726

[33] I. Letteri, G. D. Penna, L. D. Vita, and M. T. Grifa, "Mta-kdd'19: A dataset for malware traffic detection," in *CEUR, Italy, February 4-7*, M. Loreti and L. Spalazzi, Eds., vol. 2597, 2020, pp. 153–165.

[34] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[35] J. G. de la Puerta, I. Pastor-López, B. Sanz, and P. G. Bringas, "Network traffic analysis for android malware detection," in *HAIS*, ser. Lecture Notes in Computer Science, vol. 11734, 2019, pp. 468–479.

[36] A. D., V. K. K.A., S. C. S., and V. P., "Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance," *Computer Communications*, vol. 147, pp. 50–57, 2019.

[37] A. Chan and N. Vasconcelos, "Classification and retrieval of traffic video using auto-regressive stochastic processes," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, 2005, pp. 771–776.

[38] T. Zhang and C.-C. J. Kuo, "Content-based classification and retrieval of audio," in *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, F. T. Luk, Ed., vol. 3461, International Society for Optics and Photonics. SPIE, 1998, pp. 432 – 443. [Online]. Available: https://doi.org/10.1117/12.325703

[39] A. Lichy, O. Bader, R. Dubin, A. Dvir, and C. Hajaj, "When a rf beats a cnn and gru, together—a comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification," *Computers & Security*, vol. 124, p. 103000, 2023.

[40] IBM, "What is random forest?" 2023. [Online]. Available: https://www.ibm.com/topics/random-forest

[41] T. Chen, "All versus one: an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software," in *SEAMS*, 2019, pp. 157–168.

[42] H. Qi, J. Wang, W. Li, Y. Wang, and T. Qiu, "A blockchain-driven iiot traffic classification service for edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2124–2134, 2021.

[43] IBM, "K-nearest neighbors algorithm," 2023.

[44] C. Ma, X. Du, and L. Cao, "Improved knn algorithm for fine-grained classification of encrypted network flow," *Electronics*, vol. 9, no. 2, 2020.

[45] O. Kramer, *Dimensionality reduction with unsupervised nearest neighbors*. Springer, 2013, vol. 51.

[46] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," *VLDBE*, vol. 9, no. 1, pp. 1–12, 2015.

[47] elastic, "What is elastic search?" 2023. [Online]. Available: https://www.elastic.co/what-is/elasticsearch

[48] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 20 827–20 840. [Online]. Available: https://proceedings.mlr.press/v162/sun22d.html

[49] B. Duncan, "Malware traffic analysis," 2021. [Online]. Available: https://www.malware-traffic-analysis.net/

[50] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," *CoRR*, 2018.

[51] O. Bader, A. Lichy, C. Hajaj, R. Dubin, and A. Dvir, "Maldist: From encrypted traffic classification to malware traffic detection and classification," in *CCNC*. IEEE, 2022.

[52] J. Brownlee, "How to choose a feature selection method for machine learning," 2019. [Online]. Available: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[53] A. Weizmann and R. Buskila, "Feature selection." [Online]. Available: https://github.com/ComputerScienceMasterStudent/EncryptedTrafficClassification

[54] T. Jeong and H. Kim, "Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3907–3916, 2020.

[55] H. Mariuse, "Tree algorithms explained: ball tree algorithm vs. kd tree vs. brute force," 2020. [Online]. Available: https://towardsdatascience.com/tree-algorithms-explained-ball-tree-algorithm-vs-kd-tree-vs-brute-force-9746debcd940

[56] D. M. Mount, "Ann programming manual," Technical report, Dept. of Computer Science, U. of Maryland, Tech. Rep., 1998.

[57] E. Bernhardsson, "New approximate nearest neighbor benchmarks," 2018.

[58] V. Garcia, E. Debreuve, F. Nielsen, and M. Barlaud, "K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 3757–3760.

[59] V. Ramasubramani, B. D. Dice, E. S. Harper, M. P. Spellings, J. A. Anderson, and S. C. Glotzer, "freud: A software suite for high throughput analysis of particle simulation data," *Computer Physics Communications*, vol. 254, p. 107275, 2020.

[60] W. Chen, J. Chen, F. Zou, Y.-F. Li, P. Lu, and W. Zhao, "Robustiq: A robust ann search method for billion-scale similarity search on gpus," in *Proceedings of the 2019 on international conference on multimedia retrieval*, 2019, pp. 132–140.

[61] M. M. Rahman and J. Tešić, "Evaluating hybrid approximate nearest neighbor indexing and search (hannis) for high-dimensional image feature search," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 6802–6804.

[62] O. Krause-Sparmann, "Ann search in large, high-dimensional data sets (in python)," 2013.

[63] Z. Zhang, X. Huang, C. Sun, S. Zheng, B. Hu, J. Varadarajan, Y. Yin, R. Zimmermann, and G. Wang, "Sextant: Grab's scalable in-memory spatial data store for real-time k-nearest neighbour search," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 243–251.

[64] D. Yan, Y. Wang, J. Wang, H. Wang, and Z. Li, "K-nearest neighbor search by random projection forests," *IEEE Transactions on Big Data*, vol. 7, no. 1, pp. 147–157, 2019.

[65] FCC, "Voice over internet protocol," 2023. [Online]. Available: https://www.fcc.gov/general/voice-over-internet-protocol-voip

[66] O. Afilabi, "What is video on demand (vod) streaming and how does it work?" 2022. [Online]. Available: https://www.makeuseof.com/what-is-video-on-demand-how-it-works/