GRANULAR NETWORK TRAFFIC CLASSIFICATION FOR STREAMING TRAFFIC USING INCREMENTAL LEARNING AND CLASSIFIER CHAIN

Faiz Zaki [1]***, Firdaus Afifi*** [2]***, Abdullah Gani*** [3]***, Nor Badrul Anuar*** [4*]

[1,4]Department of Computer System and Technology, Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia

[2]Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Kuala Nerus, 21030 Terengganu, Malaysia

[3] Faculty of Computing and Informatics, University Malaysia Sabah, International Campus Labuan, Sabah, Malaysia

Email: faizzaki@um.edu.my[1], fdaus.isa@gmail.com[2], abdullahgani@ums.edu.my[3], badrul@um.edu.my[4*](corresponding author)

DOI: https://doi.org/10.22452/ mjcs.vol35no3.5

ABSTRACT

In modern networks, network visibility is of utmost importance to network operators. Accordingly, granular network traffic classification quickly rises as an essential technology due to its ability to provide high network visibility. Granular network traffic classification categorizes traffic into detailed classes like application names and services. Application names represent parent applications, such as Facebook, while application services are the individual actions within the parent application, such as Facebook-comment. Most studies on granular classification focus on classification at the application name level. Besides that, evaluations in existing studies are also limited and utilize only static and immutable datasets, which are insufficient to reflect the continuous and evolving nature of real-world traffic. Therefore, this paper aims to introduce a granular classification technique, which is evaluated on streaming traffic. The proposed technique implements two Adaptive Random Forest classifiers linked together using a classifier chain to simultaneously produce classification at two granularity levels. Performance evaluation on a streaming testbed setup using Apache Kafka showed that the proposed technique achieved an average F1 score of 99% at the application name level and 88% at the application service level. Additionally, the performance benchmark on ISCX VPN non-VPN public dataset also maintained comparable results, besides recording classification time as low as 2.6 ms per packet. The results conclude that the proposed technique proves its advantage and feasibility for a granular classification in streaming traffic.

Keywords: Network Traffic Classification, Streaming Traffic, Network Management, Incremental Learning, Classifier Chain, Granular, Encrypted

1.0 INTRODUCTION

According to a recent white paper by Huawei, the giant technology company forecasted that network operators would experience a substantial hike up to ten times in network traffic volume by 2025 [1]. Similarly, Cisco echoed the statement by highlighting the explosive growth of global IP traffic, which had grown 189% from 96.1 Exabytes per month in 2016 to 278.1 Exabytes per month in 2021 [2]. Additionally, the same report also forecasted the number of next-generation applications to hit nearly 300 billion by 2023, effectively changing the network landscape in the years to come. Consequently, network management tasks, particularly monitoring and security, become more complex for network operators. For example, network operators need to have solid apprehension of the network traffic flowing through their networks to carry out appropriate actions, such as quality-of-service (QoS) and security implementation. The efficacy of these actions largely depends on the network visibility level acquired by network operators. In other words, as the network visibility increases, network operators gain a significant advantage to manage their network at a more granular level. Network operators acquire network visibility through a process known as network traffic classification. Network traffic classification is extremely crucial in modern networks as it allows network operators to know the original application which generated any particular traffic. Due to its crucial importance, there have been numerous research efforts to produce network traffic classifiers using various innovative techniques [3, 4].

Among the available network traffic classification techniques include using the port and payload (i.e., deep packet inspection (DPI)) information [5]. Although these techniques have succeeded in the past, their effectiveness is continuously diminishing in modern networks. For example, dynamic port allocation leads to non-standard port assignment, making the port-based technique obsolete on its own [6]. At the same time, the increasing adoption of encrypted protocols, such as the transport layer security (TLS) protocol, encrypts the payload, reducing the effectiveness of DPI solutions. As a result, machine learning classifiers emerge as the most promising technique to overcome these challenges [7-9]. This is due to its capability to intelligently learn and generalize the model to fit various use cases relating to the network domain [10, 11]. Machine learning classifiers mostly utilize statistical properties from network packets or flows, such as the average packet and payload sizes, as input features to discriminate between different traffic classes [12]. For instance, Cao, et al. [8] and Gómez, et al. [13] utilized support vector machines and decision trees, while Lotfollahi, et al. [14] utilized the more advanced deep learning algorithm to classify network traffic into various classes. Despite achieving superior performance, existing studies portray several limitations, especially in classification granularity and strategy.

Classification granularity generally consists the coarse (i.e., application protocol and type) and fine-grained (i.e., application name and service) classification [15]. Although recent studies have started focusing on fine-grained classification, most focus on classifying traffic only at the application name level instead of application service [14, 16]. Application name classification is becoming insufficient to support today's increasingly complex network, with numerous applications offering multiple services, such as chatting and video streaming. On the other hand, the classification strategy considers offline and online classification. Most studies only consider offline classification, which utilizes readily available static data to evaluate the proposed techniques. Although efforts on offline classification have shown tremendous progress, they fail to address the practical limitations, such as implementation feasibility in a more realistic online classification setting. Indeed, it is more challenging to classify a continuous stream of network traffic with high volume and velocity than static data [17].

Therefore, this paper addresses the two limitations mentioned above by proposing an innovative technique to classify streaming network traffic with the highest granularity. To classify network traffic with the highest granularity, the proposed technique improvises on the classifier chain approach, which is well known for multi-label classification. It is also worth noting that this paper extends from our earlier study, utilizing the classifier chain but with batch learning and static data [18]. A classifier chain links multiple classifiers, with each classifier incrementally extending its input feature space by taking the output of the preceding classifier, effectively maintaining label interdependency between classifiers. Label interdependency is important as it narrows down the decision space when classifying similar services (e.g., video streaming) from different applications, such as Facebook and Youtube. In this paper, the classifier chain links two incremental classifiers, which are the App and Service-Classifier, to classify streaming traffic at both application name and service levels. Additionally, the incremental classifier implements the adaptive random forest (ARF) algorithm, which allows it to learn and classify continuously whenever new network packets arrive (i.e., streaming). As a result, it removes the need to store any static data, as demonstrated in existing studies. Furthermore, this paper also utilizes a highly scalable event streaming platform, Apache Kafka, to handle streaming traffic more efficiently.

To evaluate the performance of the proposed technique at both granularity levels, this paper used the ISCX VPN-nonVPN public dataset [19] and private ground truth. The ground truth process collected the traffic of 43 different application services across ten applications from four different locations: a campus network and three home networks. The ground truth collection process spanned six months using Grano-GT, a specialized tool to create a reliable granular ground truth [20]. Additionally, the experiments conducted in this paper replayed the dataset using *tcpreplay* [21] to simulate streaming traffic. The prequential evaluation showed that the proposed incremental technique achieved more than 88% average F1 scores across all granularity levels. In addition, this paper also benchmarked the performance against a conventional non-hierarchical classifier (i.e., a flat classifier) and the public dataset. In summary, the main contributions of this paper are as follows:

a) A focused performance evaluation on streaming traffic classification to reflect a more realistic scenario in a real-world deployment.
b) A new approach to classify streaming traffic at the application name and service levels using incremental learning and classifier chain.

The remainder of the paper is organized as follows. Section 2 discusses the related works and our contribution positioning. Section 3 discusses the proposed technique and provides comprehensive interpretations of the overall architecture. Section 4 outlines the experimental analysis, and Section 5 discusses the current issues in the domain and how to move forward. Finally, Section 6 concludes the paper.

2.0 RELATED WORK AND CONTRIBUTION POSITIONING

Network traffic classification has been in the constant spotlight over the last decade. As such, there are numerous works to produce the most accurate and reliable network traffic classifier. This section presents a comprehensive overview of existing and related works focusing on network traffic classification using incremental learning and streaming traffic. This section also outlines our contributions to highlight the key improvements compared to existing works.

2.1 Network Traffic Classification Using Incremental Learning

Among the drawbacks of conventional machine learning paradigms, such as supervised and unsupervised learning, is the assumption of a complete and readily available dataset. However, a network traffic dataset is seldom complete as it is continuous and infinite by nature. Therefore, incremental learning emerges as a promising approach. It removes the need for a complete training dataset and enables the classifier to adapt to evolving network traffic through partial retraining in the future. Despite that, the implementation of incremental learning for network traffic classification is scarce. For example, Loo and Marsono [22] proposed the incremental k-means algorithm to classify network traffic into coarse-grained classes with more than 90% accuracy even when utilizing only 10% of labelled input flows. The authors also evaluated the proposed technique with streaming traffic and classified 25 thousand flows per second. Despite the excellent results, the proposed technique only considered application protocol and type granularity, which is insufficient in modern networks.

Besides that, Sun, et al. [23] and Punitha and Mala [24] utilized the incremental adaptation of support vector machines (SVM). Incremental SVM largely reduces the algorithm complexity by maintaining the prior support vectors during retraining, avoiding a full retraining procedure. In [23], the authors further improvised the initial implementation by introducing an attenuation factor to retain valuable information from prior training data and recorded more than 90% average accuracy and significant speedup compared to conventional SVM. On the other hand, the authors in [24] focused on detecting network attacks, such as denial of service. However, both discussed works were also limited to coarse-grained granularity despite demonstrating superior performances. Realizing the need for granular classification, Bovenzi, et al. [25] and Chen, et al. [26] proposed incremental learning using deep learning algorithms to classify traffic at the application name level. Another crucial advantage of deep learning algorithms is that they provide automatic feature representation, eliminating the reliance on manually handcrafted features in conventional machine learning algorithms. Besides that, [25] and [26] also demonstrated class incremental learning, allowing easy adding of new and unseen classes. However, both studies showed a steady decline in performance as new classes increased and only focused on mobile traffic classification.

2.2 Streaming Network Traffic Classification

Most of the proposed techniques mentioned in the previous section utilized static and immutable datasets for evaluation. Indeed, besides the scarcity of incremental learning for network traffic classification, evaluating network traffic classifiers using streaming traffic also receives little attention and remains a challenge [27]. For example, besides the work in [22], Carela-Español, et al. [17] was the only work that focused on using incremental learning and evaluated it in a stream setting. The authors utilized the prequential evaluation approach and achieved more than 95% average accuracy. Prequential evaluation measures the performance of streaming algorithms by monitoring the learning evolution over time [28]. In brief, the approach uses the data to test the model first before reusing the same data to train the model. It closely resembles the stream setting where an incremental learner classifies the newly arrived and unseen data before using them to carry out the training process. Similarly, this paper utilized the prequential evaluation approach to measure the performance of the proposed technique. However, this paper also simulated streaming traffic using *tcpreplay* to allow variations in the streaming speed, allowing a more thorough evaluation.

On the other hand, Labayen, et al. [29] proposed a hierarchical classifier consisting of two layers of k-means classifier and a random forest layer. Each layer corresponds to hierarchical windows proposed by the authors, which are behavioural, flow and classification. These windows allowed the authors to filter multiple user activities in network traffic. The proposed technique recorded 97% average accuracy with a classification time of 50 ms per classification window. Similarly, our work also proposed a hierarchical classifier by utilizing a classifier chain to achieve a more granular classification at the application name and service levels. In [30], the authors proposed using a convolutional neural network (CNN) to classify streaming traffic. The proposed technique recorded an impressive classification time at approximately 2 ms per packet with more than 90% F1 scores across multiple granularity levels (i.e., protocol, type and name). However, both [29] and [30] used non-incremental algorithms, thus facing limitations in addressing evolving network traffic with progressive training.

Table 1 summarizes the related works discussed previously based on several criteria, such as whether they utilized incremental learning and evaluated streaming traffic, the number of features, classifier, number of output classes and output granularity level. Moreover, Table 1 also compares the differences between the related works and this paper, thus highlighting its significance. For example, most works that utilized incremental learning only produced coarse-grained classification. This paper extends the output granularity achieved through incremental learning by producing a more granular classification (i.e., at the application name and service levels) to address the increasing need for network visibility in modern networks. On the other hand, based on Table 1, there is also a clear lack of attention for streaming traffic evaluation and even more so for works that combine incremental learning and streaming traffic evaluation. This paper takes advantage of such a gap to propose a solution that combines both factors to produce a more practical network traffic classification solution that closely represents a real-world use case.

Table 1: Summary of related works

Columns and acronyms are as follows: *Incremental learning (IL). Feature dimension (FD). Classifier;* + sign indicates classifier combinations. *Number of output classes (NC).*

Reference	IL	Streaming	FD	Classifier	NC	Granularity
[22]	✓	✓	11	Incremental k-means	10	Protocol, Type
[23]	✓	×	256	Incremental SVM	11	Type
[24]	✓	×	9	Incremental hybrid (SVM + Unsupervised classifier)	2	Protocol
[25]	✓	×	N/A	Incremental deep learning	40	Name
[26]	✓	×	N/A	Incremental OnevRest neural network	16	Name
[17]	✓	✓	16	Hoeffding Adaptive Tree	10	Protocol
[29]	×	✓	11	Multi-layer classifier (k-means + random forest)	5	Type
[30]	×	✓	N/A	Convolutional neural network	6	Protocol, Type, Name
This Paper	✓	✓	7	Adaptive Random Forest with classifier chain	43	Name, Service

2.3 Contribution Positioning

Accordingly, we position our work against similar works in the domain discussed above as follows:

- This paper focuses on the most granular classification levels, which are application name and service. The granular classification is crucial to address the increasingly complex modern network that requires more network visibility.
- This paper proposes an innovative technique that handles streaming network traffic instead of readily available static datasets to represent a more realistic scenario in a real-world deployment. The proposed technique also utilizes the prequential evaluation approach to ensure a fair performance evaluation.
- This paper takes advantage of the classifier chain to produce a multi-label classification and maintain label interdependency between classifiers. Maintaining label interdependency offers a significant advantage in narrowing down the decision-making when classifying similar services from different applications.
- This paper utilizes a novel feature set based on payload length statistics to discriminate traffic at the application name and service levels. The improved feature set includes moving statistics (i.e., moving averages) that capture the changing dynamics in fine-grained traffic flows.
- This paper validates the proposed technique on a self-collected dataset and the well-known ISCX VPN-nonVPN public dataset. Using a public dataset allows for a fair evaluation to prove the effectiveness of our work.

3.0 METHODOLOGY

In this paper, our proposed technique implements two Adaptive Random Forest (ARF) classifiers linked together using a classifier chain to produce a granular network traffic classification at the application name and service levels. An ARF classifier is a reliable incremental learner which improvises from the conventional random forest and Hoeffding tree. Additionally, this paper evaluates the classifier on a streaming testbed using Apache Kafka. This section breaks our implementation into three main phases: dataset and feature extraction, model design with ARF and classifier chain and model evaluation on Apache Kafka to better elaborate our methodology. Fig 1 illustrates the phases as a workflow where each phase builds progressively to serve the subsequent phases. For example, the first phase outputs a base feature set that serves as the second phase's input.

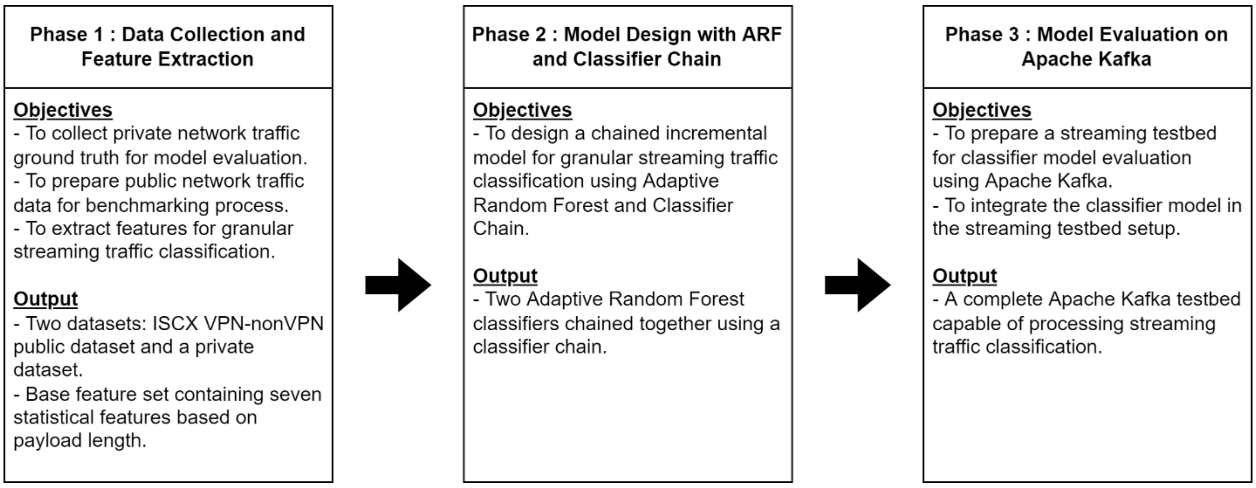
Phase 1 : Data Collection and Feature Extraction
Objectives
- To collect private network traffic ground truth for model evaluation.
- To prepare public network traffic data for benchmarking process.
- To extract features for granular streaming traffic classification.
Output
- Two datasets: ISCX VPN-nonVPN public dataset and a private dataset.
- Base feature set containing seven statistical features based on payload length.
Phase 2 : Model Design with ARF and Classifier Chain
Objectives
- To design a chained incremental model for granular streaming traffic classification using Adaptive Random Forest and Classifier Chain.
Output
- Two Adaptive Random Forest classifiers chained together using a classifier chain.
Phase 3 : Model Evaluation on Apache Kafka
Objectives
- To prepare a streaming testbed for classifier model evaluation using Apache Kafka.
- To integrate the classifier model in the streaming testbed setup.
Output
- A complete Apache Kafka testbed capable of processing streaming traffic classification.

Fig 1: Workflow of the proposed methodology

3.1 Data collection and Feature Extraction

A reliable dataset is crucial for a fair and trustworthy classifier performance evaluation. Therefore, this paper utilizes two datasets, namely the public ISCX VPN-nonVPN dataset and our private dataset. The ISCX VPN nonVPN dataset is the most used dataset in recent studies [29, 30] and is accessible by request to the authors. It is also among the most updated dataset in capture time and granularity levels, making it suitable for this paper. The dataset contains raw network traffic captured in a controlled environment, in which synthetic user accounts were created to generate traffic from target applications based on predetermined tasks. In addition, the dataset preparation process utilized an isolated environment to ensure reliability by shutting unnecessary services and filtering only packets bearing the client’s source or destination IP. The 28 GB dataset contains two main categories: traffic with standard encryption and encrypted traffic tunnelled through a VPN. Table 2 shows the dataset contents, in which both categories have seven application types: browsing, email, chat, streaming, file transfer, VoIP and P2P, totalling 14 traffic classes for both VPN and non-VPN settings.

Table 2: ISCX VPN-nonVPN dataset

Application type	Application name
Browsing	Firefox and Chrome
Email	SMPTPS, POP3S and IMAPS
Chat	ICQ, AIM, Skype, Facebook and Hangouts
Streaming	Vimeo and Youtube
File Transfer	Skype, FTPS and SFTP using Filezilla and an external service
VoIP	Facebook, Skype and Hangouts voice calls
P2P	uTorrent and BitTorrent

Besides the public dataset, this paper also uses a private dataset containing raw browser-based traffic from 43 different application services across ten applications. The capture process utilized Grano-GT, a specialized tool developed to ensure the highest reliability in the collection process spanning six months, from July 2020 to January 2021. In addition to temporal differences, the dataset also introduced spatial differences by capturing the traffic at four different locations, namely three home networks and a campus network. Table 3 lists the different applications and services in the dataset, including other details such as the network environment and the generic type. All the datasets that make up the private dataset contain the same applications as listed in the application name and service column. For example, the USJ-20 dataset was collected at a private 300 Mbps home network and contained ground truth traffic traces from various applications and services, such as Facebook, Facebook-comment and Youtube-video.

Table 3: Private dataset

Dataset	Network	Application name and service	Application type
USJ-20	300 Mbps home network	Facebook: comment, post, video, chat, react	Social media
		Twitter: tweet, retweet, upload, video, react	Social media
		Youtube: browse, comment, upload, video, react	Video streaming
KCS-20	30 Mbps home network	Netflix: video, react	Video streaming
		Lazada: browse, buy, chat, react	eCommerce
		Shopee: browse, buy, chat, react	eCommerce
AI-20	30 Mbps home network	Telegram: chat, image, document, video, audio	Messaging
		Web-Whatsapp: chat, image, document, video, audio	Messaging
WISMA-20	100 Mbps campus network	Medium: browse, comment, post, react	Publishing
		Reddit: browse, comment, post, react	Publishing

Another important factor that contributes to the classifier performance is the input features used to discriminate between different traffic classes. Table 4 lists the base feature set with the respective details for each feature. The base feature set contains seven lightweight features based on payload length. Using features based on payload length maintain user privacy as they avoid any payload content inspection. Also, the base feature set requires 100 packets at most. Although 100 packets seem relatively large, it produced the best performance considering the complexity of distinguishing fine-grained traffic. In the event of a flow containing less than 100 packets, the proposed technique still computes the feature statistics, however potentially resulting in sub-optimal performance. Therefore, this paper considers the 100-packet requirement a slight limitation in the proposed technique.

Table 4: Base feature set

No.	Feature	Description
1	protocol	Layer 4 protocol, i.e., TCP or UDP
2	max_avg_payload	The maximum of average payload length in either direction, i.e., source to destination or vice versa
3	mss_count_100	The count of packets in the first 100 having the payload length equalling the maximum segment size
4	range_10	The range of payload length for the first ten packets, i.e., maximum – minimum
5	std_10	The standard deviation of payload length for the first ten packets
6	ma_5	The five-packet moving average for payload length
7	ma_40_avg_5	The average of the first five entries of 40-packet moving average for payload length

3.2 Model Design with ARF and Classifier Chain

The base feature set listed in Table 4 serves as the input feature to two ARF classifiers: the App-Classifier and Service-Classifier. In order to classify at two granularity levels simultaneously, this paper utilizes the classifier chain to link both classifiers together. In brief, a classifier chain appends the output from the App Classifier to the initial base feature set and becomes the input for the Service Classifier. A classifier chain is ideal for producing an optimal granular multi-label classification as it maintains the label dependency between the application names and services. Additionally, the proposed framework modifies the original implementation of a classifier chain [31]. The proposed technique chains only two ARF classifiers, equalling the target granularity levels (i.e., application name and service) instead of creating binary classifiers based on the total traffic classes as in the original implementation. Fig 2 illustrates the classifier chain implementation, highlighting the feature space extension to introduce the label dependency in the Service-Classifier. In more detail, the input of Classifier 2 contains the initial features from Classifier 1 together with its output as an extended feature.

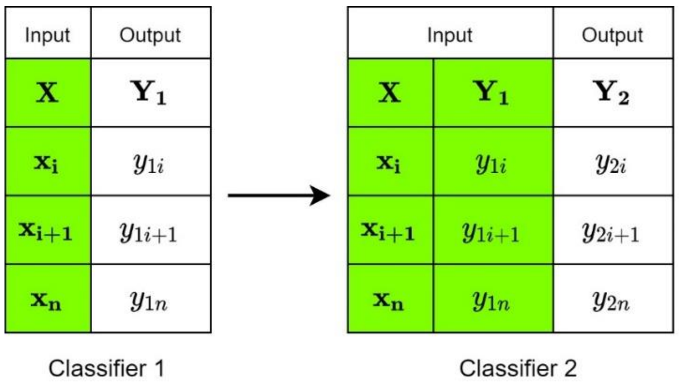

Fig 2: Proposed classifier chain implementation

3.3 Model Evaluation on Apache Kafka

Furthermore, we build our testbed setup using Apache Kafka to simplify evaluation in a stream setting. Apache Kafka is an open-source distributed event streaming tool for efficient stream processing through its topic-partition architecture. Apache Kafka processes streaming data by organizing them into topics. A topic is a user-defined collection that scales efficiently by using partitions. These partitions are capable of spanning across multiple machines, making them ideal for real-time processing.

In general, the Apache Kafka setup consists of four main modules: Stream Producer, Flow Broker, Flow Consumer and Feature Extractor and Classification Broker. Fig 3 shows the overall architecture of the proposed technique. The Stream Producer accepts the input data (i.e., PCAP files) and utilizes *tcpreplay* and PyShark for streaming and preprocessing. Tcpreplay takes PCAP files from the dataset and replays them through the network interface in the exact order to simulate live network traffic flow. Then, PyShark captures the replayed traffic and extracts the payload length and protocol (i.e., the raw features) and 5-tuple information. Both tcpreplay and PyShark serves as the message producer to feed the Flow Broker. The Flow Broker accepts the extracted raw features and 5-tuple as streams and efficiently splits them into separate partitions based on the 5-tuple. In other words, the Flow Broker effectively extracts the flows from the streaming traffic and passes the flows to the Flow Consumer.

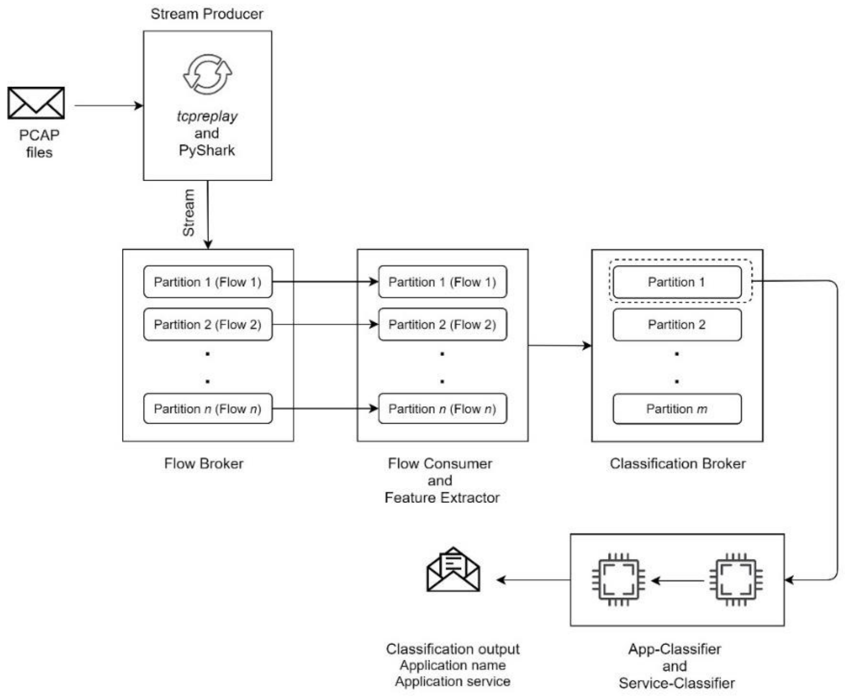

Fig 3: Overall architecture of Apache Kafka testbed

The Flow Consumer receives the flows from each partition. The number of partitions in the consumer equals the previous broker. Therefore, the Flow Consumer handles each partition (i.e., flow) separately, thus maintaining perfect isolation for further flow processing. In the processing phase, the consumer rapidly extracts the base feature set, as discussed earlier. Once done, it passes the extracted features, which is now ready for classification to the Classification Broker. The Classification Broker consists of multiple partitions that represent separate classification processes. All classification processes generated by the partitions go through the App-Classifier and Service classifier mentioned in the previous section to produce the granular classification output at the application name and service levels.

4.0 RESULTS AND DISCUSSION

This paper conducted two main experiments to evaluate the proposed approach. Namely, the first experiment evaluated the proposed technique when classifying traffic using private datasets. We combined all the private datasets to introduce spatial and temporal variabilities in the data. In addition, this paper also compared the classification performance with the baseline classifiers. The second experiment focused on evaluating the robustness of the proposed technique when tested on a public dataset. Furthermore, the evaluation also includes measuring the proposed technique's performance in terms of its latency and classification time in a stream setting.

4.1 Performance on private dataset

Regular classifier updates are critical to ensure that the classifier remains reliable over time. Therefore, this paper compared the App-Classifier and a conventional batch learner (i.e., random forest) performance over multiple updates using the prequential evaluation technique. The technique divides the dataset into batches and attempts to classify them before reusing them to train the model. This experiment tested various batch sizes (i.e., 1000, 10000 and 100000 packets) to analyze the effects on both classifiers' performance. Fig 4 illustrates the effect of different batch sizes on the F1 scores using prequential evaluation. The horizontal axis represents the packet counts, while the vertical axis represents the corresponding F1 scores. From the figure, the incremental App-Classifier recorded a lower F1-score in the first iteration (i.e., cold start) but steadily increased and maintained F1-scores higher than 0.9 throughout the evaluation. On the other hand, the batch learning random forest model recorded decreasing and inconsistent F1-scores. This is because batch learning creates a new model from scratch in each iteration, dismissing all prior knowledge. In contrast, incremental learning retains prior knowledge and updates progressively. However, the random forest model demonstrated improved performance when using larger batch sizes. Larger batch sizes allowed the random forest model to learn more information about the data, hence recording better F1-scores. Unlike the random forest, the App-Classifier showed marginal impact on varying batch sizes, thus omitted from Fig 4. Despite that, the limitation with batch learning still holds, which explains its inability to surpass App-Classifier in the evaluation. The result demonstrated a clear advantage of incremental learning (i.e., knowledge retention), which corresponds closely to a real-world use case where the network traffic constantly evolves and requires periodic updates.

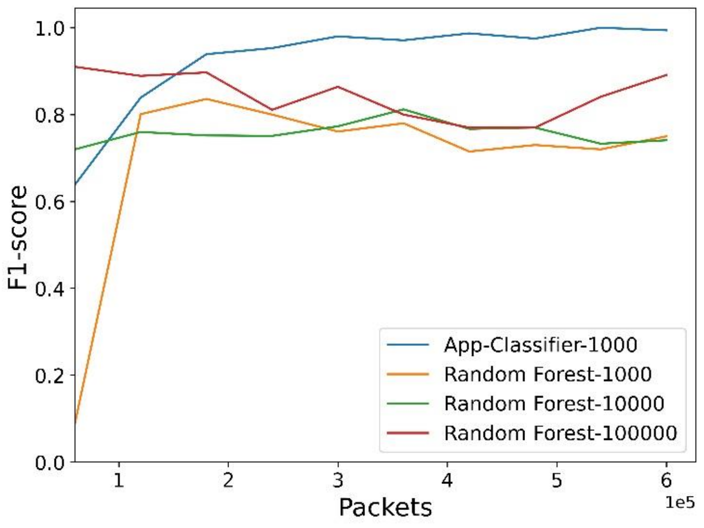

Fig 4: Prequential evaluation on App-Classifier and Random Forest by batch size

Table 5 dives into details on the classification results per application for App-Classifier. Based on the results, App-Classifier demonstrated superior performance at the application name level, achieving perfect accuracy. Generally, we expected this outcome as the distinct characteristics of different applications were significant. In other words, application classification is a more straightforward task because each application is considerably unique compared to higher granularity classification, such as at the application service level. However, we observed a fairer classifier evaluation using precision, recall, and F1-score metrics. For example, Shopee, a well-known eCommerce application in the Southeast Asia region, recorded lower recall when compared to its precision, indicating higher false negatives than false positives. Nonetheless, the recorded performance strongly instantiates the proposed technique's strength, including the payload-based features and classifier, to classify at the application name level.

Table 5: App-Classifier classification results per application

Application	**Precision**	**Recall**	**F1-score**
Facebook	1.00	1.00	1.00
Web-Whatsapp	1.00	1.00	1.00
Telegram	0.99	0.99	0.99
Lazada	1.00	1.00	1.00
Shopee	1.00	0.91	0.95
Twitter	1.00	1.00	1.00
Youtube	1.00	1.00	1.00
Medium	1.00	1.00	1.00
Reddit	1.00	1.00	1.00
Netflix	1.00	1.00	1.00
Macro Average	1.00	0.99	0.99

On the other hand, Table 6 details Service-Classifier performance per application service. The classification process in this stage was more complex than application name classification due to higher granularity and feature similarity between the services. As such, Table 6 portrays the effect of such complexity where the Service-Classifier recorded a lower overall F1-score at 88%, compared to App-Classifier previously. Table 6 also details the macro average per application to highlight application services that were more complex to classify. For example, Youtube recorded lower than average scores, especially with Youtube-react. Besides that, Shopee, a leading eCommerce application in the Southeast Asia region, also performed poorly. All intra-application services within Shopee recorded F1-scores lower than the overall average. Further analysis indicated that some applications utilized the same network flow to carry multiple intra-application services. As a result, it affected the classification performance because most of the input features were flow-related. For example, based on our experiments, Facebook-comment and Facebook-post were generated from a single network flow. Accordingly, Szabó, et al. [32] also highlighted that a traffic flow could be used for multiple purposes throughout its lifetime, echoing our findings mentioned previously. Despite that, Service-Classifier still managed to produce a commendable performance at the application service level, considering the low complexity of the proposed features.

Table 6: Service-Classifier classification results per application service

Applications	**Service**	**Precision**	**Recall**	**F1-score**
Facebook	comment	0.43	0.23	0.30
	post	0.82	0.92	0.87
	video	1.00	1.00	1.00
	chat	1.00	1.00	1.00
	react	1.00	1.00	1.00
Macro Average		0.85	0.83	0.83
Web-Whatsapp	chat	1.00	1.00	1.00
	image	0.98	0.93	0.95
	document	1.00	0.99	0.99
	audio	1.00	1.00	1.00
	video	1.00	1.00	1.00
Macro Average		0.99	0.98	0.98
Telegram	chat	1.00	0.86	0.93
	image	0.89	1.00	0.94
	document	0.92	1.00	0.96
	audio	1.00	0.90	0.95
	video	1.00	0.99	0.99
Macro Average		0.96	0.95	0.95

Lazada	browse	1.00	1.00	1.00
	buy	1.00	1.00	1.00
	chat	1.00	0.92	0.96
	react	1.00	1.00	1.00
Macro Average		1.00	0.98	0.99
Shopee	browse	0.75	1.00	0.86
	buy	1.00	0.25	0.40
	chat	0.60	0.67	0.63
	react	0.73	1.00	0.84
Macro Average		0.77	0.73	0.75
Twitter	tweet	1.00	1.00	1.00
	retweet	0.60	0.58	0.59
	upload	1.00	0.98	0.99
	video	1.00	1.00	1.00
	react	1.00	1.00	1.00
Macro Average		0.92	0.91	0.91
Youtube	browse	1.00	1.00	1.00
	comment	0.51	0.75	0.61
	upload	1.00	1.00	1.00
	video	1.00	1.00	1.00
	react	0.44	0.22	0.29
Macro Average		0.79	0.79	0.79
Medium	browse	1.00	0.93	0.96
	comment	0.50	0.33	0.40
	post	0.78	0.84	0.81
	react	1.00	1.00	1.00
Macro Average		0.82	0.78	0.79
Reddit	browse	1.00	1.00	1.00
	comment	0.90	0.95	0.93
	post	0.76	0.70	0.73
	react	0.86	1.00	0.92
Macro Average		0.88	0.91	0.89
Netflix	react	1.00	1.00	1.00
	video	1.00	1.00	1.00
Macro Average		1.00	1.00	1.00
Overall Macro Average		0.89	0.88	0.88

4.2 Performance on public dataset

The ISCX VPN-nonVPN dataset is the most widely used in recent works. The evaluation combined application services from the same application in the dataset in order to allow evaluation at the application name level. For example, the evaluation combined Facebook-audio, Facebook-chat and Facebook-video into a single Facebook application. Table 7 shows the classification results. App-Classifier achieved a 0.98 average F1-score, recording F1-scores higher than 0.8 for all individual applications. The AIM application recorded a 0.89 F1-score, which was the lowest score in the evaluation. The result was within expectation as AIM constituted less than 1% of the dataset, thus providing little information to App-Classifier during the training process.

Table 7: App-Classifier classification results on ISCX VPN-nonVPN dataset

Applications	Precision	Recall	F1-score
AIM	0.88	0.90	0.89
Email	1.00	1.00	1.00
Facebook	0.98	0.97	0.98
Gmail	1.00	1.00	1.00
Hangouts	1.00	1.00	1.00
ICQ	0.96	0.93	0.95
Netflix	1.00	1.00	1.00
Skype	1.00	1.00	1.00
Spotify	1.00	1.00	1.00
Vimeo	1.00	1.00	1.00
Youtube	1.00	0.99	1.00
Macro Average	0.98	0.98	0.98

Similarly, the Service-Classifier also maintained the superior performance achieved by the App-Classifier when evaluated at the application service level. Table 5.24 presents the detailed classification results for the proposed approach. Service-Classifier recorded perfect classification for ten out of 14 application services. In addition, the F1 scores for most application services were more than 0.95, except AIM-chat, which caused an anomaly in the F1-score range. AIM-chat obtained an inferior recall score of 0.07, indicating substantial false negatives. Further inspection revealed that AIM-chat had the lowest samples in the ISCX VPN-nonVPN dataset, causing the classifier to underfit the class considerably.

Table 8: Service-Classifier’s classification results on ISCX VPN-nonVPN dataset

Applications	Precision	Recall	F1-score
AIM-chat	1.00	0.07	0.14
Email	1.00	1.00	1.00
Facebook-audio	1.00	1.00	1.00
Facebook-chat	1.00	0.96	0.98
Facebook-video	1.00	1.00	1.00
Gmail-chat	0.98	1.00	0.99
Hangouts-audio	0.97	1.00	0.98
Hangouts-chat	1.00	1.00	1.00
ICQ-chat	1.00	1.00	1.00
Netflix	1.00	1.00	1.00
Skype-chat	1.00	1.00	1.00
Spotify	1.00	1.00	1.00
Vimeo	1.00	1.00	1.00
Youtube	1.00	1.00	1.00
Macro Average	1.00	0.93	0.94

Fig 5 shows the confusion matrix for the proposed Service-Classifier, highlighting the false negatives in the classification. For example, Facebook-chat was slightly misclassified as Gmail-chat, explaining the recall and precision scores for Facebook-chat and Gmail-chat, respectively. Furthermore, the confusion matrix clearly shows how the Service-Classifier misclassified the majority of AIM-chat samples as Hangouts-audio (i.e., darker purple-shaded box at predicted class hangouts-audio) due to underfitting.

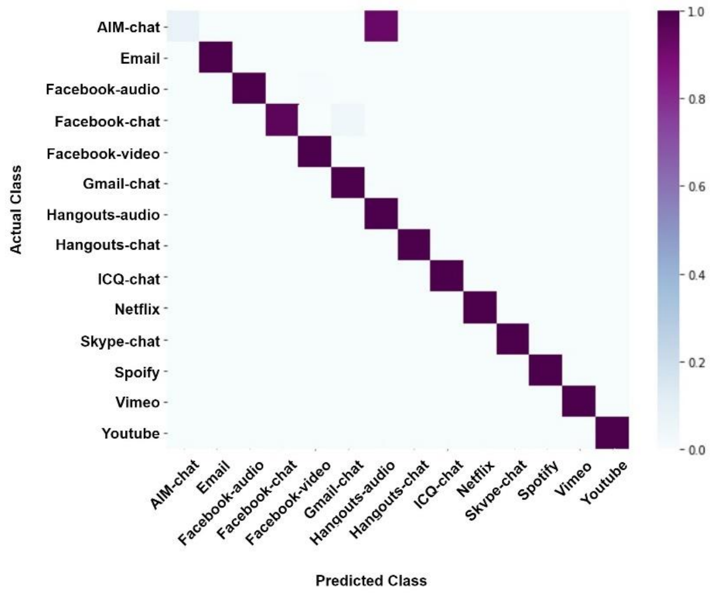

Fig 5: Confusion matrix for Service-Classifier on ISCX VPN-nonVPN dataset

4.3 Latency and Classification Time

The request latency represents the time delay between the Stream Producer and brokers. The evaluation utilized five streaming speeds: 10, 30, 100, 500 and 1000 Mbps, based on common speeds offered by local ISPs. In addition, the evaluation also considered different partition sizes to analyze their effect on the latency. Table 9 lists the request latencies at different speeds and partition sizes, with the lowest recorded times for each partition in bold. The request latency ranged from 700 ms to 1000 ms. Interestingly, although the initial expectation was to observe higher latencies with increasing streaming speeds, the opposite situation occurred. Based on the table, the average latencies across all partition sizes decreased slightly with faster streaming speeds, except at 1000 Mbps. Despite that, the latencies were generally consistent at all streaming speeds. The consistency came as an advantage as the increasing speeds had minimal effect on the request latencies. Hence, it shows the potential and feasibility of the proposed technique to operate in high-speed networks.

Table 9: Request latencies

Speed (Mbps)	**Latency (ms)**					**Average**
	Partition size					
	1	**5**	**10**	**20**	**40**	
10	900	931	963	1000	981	955
30	**856**	**863**	913	913	960	901
100	860	873	**851**	**870**	893	869
500	876	870	870	1000	**716**	**866**
1000	889	986	900	982	997	951

On the other hand, smaller partition sizes generally exhibited lower request latencies. The reason for this situation was because of the higher overheads when generating more partitions. However, larger partition sizes translate to better concurrency and scalability as they allow better load distribution in Apache Kafka. Fig 6 illustrates the resulting latencies discussed above for better comprehension. From the figure, the request latency exhibited an anomaly when using 40 partitions at 500 Mbps streaming speed. The request latency dropped to less than 800 ms, well below the 866 ms average latency. The anomaly could be due to several reasons, such as packet drops.

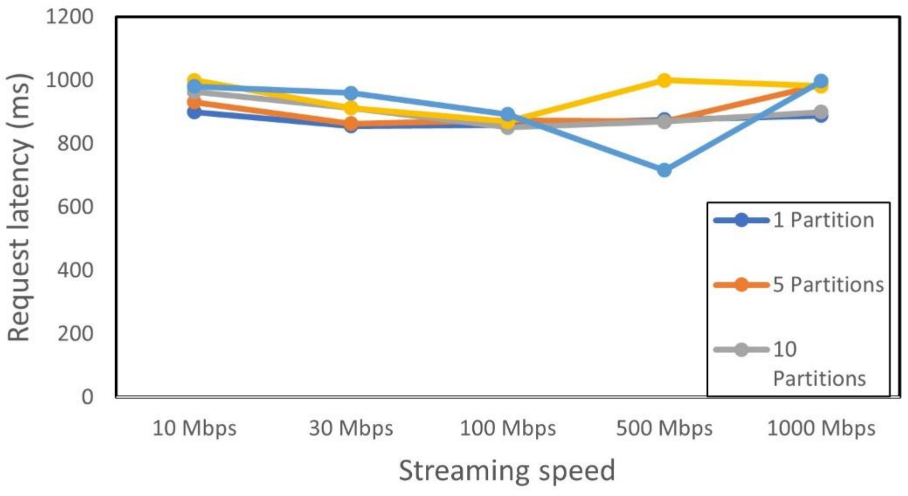

Fig 6: Request latencies at different streaming speeds

The previous evaluation highlighted how smaller partition sizes demonstrated lower request latencies. However, Apache Kafka relies on partitions to induce concurrency and scalability in load distribution. In other words, bigger partition sizes produce faster classification times as more consumers utilize the partition in parallel. Therefore, it presents an interesting trade-off between latency and classification time. The partition size in this evaluation was selected arbitrarily with a two-fold sequence starting at the second partition size. Table 10 shows the classification time for 1000 packets at different partition sizes. The table shows a significant decrease in classification time where a single partition setup recorded 98.4 seconds to classify 1000 packets while a 40 partition setup took only 2.6 seconds.

Table 10: Classification time

Partition	**Classification time (seconds)**
1	98.4
5	21.4
10	11.0
20	5.2
40	2.6

Fig 7 illustrates the results for easier understanding. It was evident that bigger partition sizes resulted in faster classification times, depicted by the exponential curve. Recalling the findings from the previous evaluation where bigger partition caused higher latencies, the evaluation found that the average cost incurred due to increased latency was 10.6%. However, in this evaluation, the improvement in classification time recorded was more than 97% (i.e., comparing percentage decrease from a single partition to 40 partitions). On average, the classification took only 2.6 ms per packet when using 40 partitions, making it feasible for real-time classification implementation. Therefore, the evaluation clearly shows the advantage of partitioning in Apache Kafka to reduce the classification time without incurring a high cost. As such, it is better to choose bigger partition sizes to improve the classification time of streaming traffic significantly.

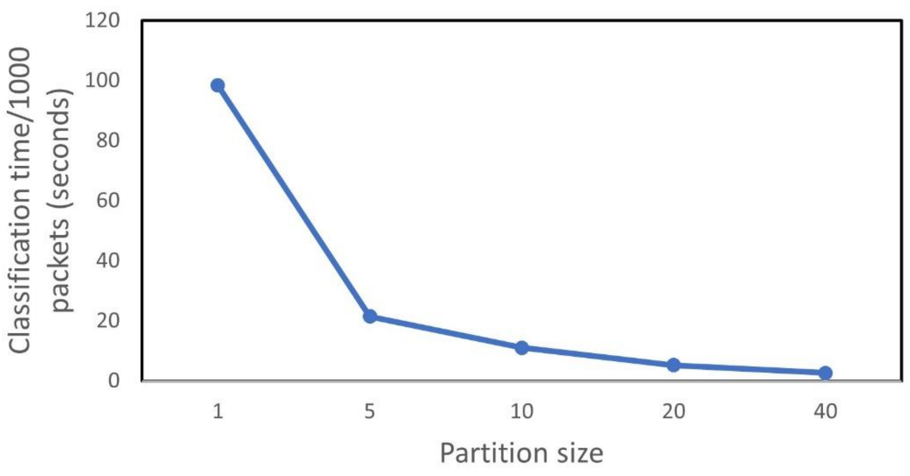

Fig 7: Classification speed on different numbers of partitions

5.0 CONCLUSION

This paper addressed the core issue in network traffic classification for modern networks where granular network traffic classification is needed while at the same time being evaluated in a streaming setting to showcase the feasibility and reliability of the proposed technique. The issue came to light as most existing works only managed to produce up to application name level classification at most, besides being evaluated using a static and immutable dataset. In response to the issue, this paper presented a technique to classify network traffic with high granularity at the application name and service levels. To achieve this objective, this paper utilized two ARF classifiers linked together using a classifier chain. The classifiers also took advantage of seven statistical features based on payload length to discriminate between applications and the different services available within the application.

Our experiments demonstrated the reliability and feasibility of our proposed technique. Evaluation on our private dataset showed that the App-Classifier achieved a 99% average F1 score at the application name level. The impressive performance reflected the effectiveness of the base feature set to differentiate between different applications accurately. On the other hand, our proposed technique also maintained commendable performance at the application service level. Interestingly, by using the extended feature set that included the output of the App-Classifier, the Service-Classifier managed to record an average F1 score of 88% despite the greater similarity between characteristics of the application services. Furthermore, we also evaluated the latency and classification time incurred at different streaming speeds. Our evaluation found that the latency generally remained consistent while recording classification time as low as 2.6 seconds per 1000 packets at higher partition sizes. On the same note, our proposed technique also achieved significant results when benchmarked against the ISCX VPN-non VPN public dataset.

However, our work is currently constrained to a limited number of applications and services because it is highly challenging to provide coverage to the massive range of available applications and services. A possible solution for future work is to explore unsupervised learning to classify unknown traffic without predefining the data. Alternatively, future works are open to improving the proposed technique with class incremental learning to add new and unseen classes easier. On the other hand, future works should also focus on producing practical real-time traffic classifiers as it is highly likely to be an invaluable technology in the future. Therefore, researchers should invest more effort in feature and algorithm optimizations to produce a highly optimized classifier capable of handling enormous real-time network traffic streams. We firmly believe the output from these researches would pave the way to many other exciting applications, such as real-time network profilers and more efficient network management systems.

6.0 ACKNOWLEDGEMENT

This work was supported by the Fundamental Research Grant Scheme under the Ministry of Education Malaysia (FRGS/1/2018/ ICT03/UM/02/3).

REFERENCES

[1] Huawei, "NetX2025 Target Network Technical White Paper," 2021, Available: https://carrier.huawei.com/~/media/CNBGV2/download/program/netx2025/netx-2025-target-network-technical-white-paper-en.pdf.

[2] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," 2020, Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, Accessed on: 25 March 2021.

[3] E. Papadogiannaki and S. Ioannidis, "A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures," *ACM Comput. Surv.*, vol. 54, no. 6, 2021.

[4] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 2, pp. 1988-2014, 2019.

[5] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Computer Networks,* vol. 76, pp. 75-89, 2015.

[6] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," presented at the Proceedings of the 6th International Conference on Passive and Active Network Measurement, Boston, MA, 2005. Available: https://doi.org/10.1007/978-3-540-31966-5_4

[7] S. Dong, "Multi class SVM algorithm with active learning for network traffic classification," *Expert Systems with Applications,* vol. 176, p. 114885, 2021.

[8] J. Cao, D. Wang, Z. Qu, H. Sun, B. Li, and C.-L. Chen, "An Improved Network Traffic Classification Model Based on a Support Vector Machine," *Symmetry,* vol. 12, no. 2, p. 301, 2020.

[9] H. Mun and Y. Lee, "Internet Traffic Classification with Federated Learning," *Electronics,* vol. 10, no. 1, 2021.

[10] N. B. Anuar, H. Sallehudin, A. Gani, and O. Zakari, "IDENTIFYING FALSE ALARM FOR NETWORK INTRUSION DETECTION SYSTEM USING HYBRID DATA MINING AND DECISION TREE," *MALAYSIAN JOURNAL OF COMPUTER SCIENCE,* vol. 21, no. 2, pp. 101-115, 2008.

[11] A. Feizollah, N. B. Anuar, R. Salleh, F. Amalina, R. R. Ma'arof, and S. Shamshirband, "A STUDY OF MACHINE LEARNING CLASSIFIERS FOR ANOMALY-BASED MOBILE BOTNET DETECTION," *MALAYSIAN JOURNAL OF COMPUTER SCIENCE,* vol. 26, no. 4, pp. 251-265, 2013.

[12] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," in "Department of Computer Science Research Reports," Queen Mary University of London2005, Available: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf.

[13] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. Hernández Callejo, "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal," *Computer Networks,* vol. 127, pp. 68-80, 2017.

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

[15] F. Zaki, A. Gani, and N. B. Anuar, "Applications and use Cases of Multilevel Granularity for Network Traffic Classification," presented at the 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), Langkawi, Malaysia, 28-29 Feb, 2020.

[16] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z. Ling, "Encrypted Network Traffic Classification Using Deep and Parallel Network-in-Network Models," *IEEE Access,* vol. 8, pp. 132950-132959, 2020.

[17] V. Carela-Español, P. Barlet-Ros, A. Bifet, and K. Fukuda, "A streaming flow-based technique for traffic classification applied to 12 + 1 years of Internet traffic," *Telecommunication Systems,* vol. 63, no. 2, pp. 191-204, 2016.

[18] F. Zaki, F. Afifi, S. Abd Razak, A. Gani, and N. B. Anuar, "GRAIN: Granular multi-label encrypted traffic classification using classifier chain," *Computer Networks,* vol. 213, p. 109084, 2022/08/04/ 2022.

[19] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," presented at the Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), 2016.

[20] F. Zaki, A. Gani, H. Tahaei, S. Furnell, and N. B. Anuar, "Grano-GT: A granular ground truth collection tool for encrypted browser-based Internet traffic," *Computer Networks,* vol. 184, p. 107617, 2021/01/15/ 2021.

[21] F. Klassen and AppNeta. (2021). *Tcpreplay - Pcap editing and replaying utilities*. Available: https://tcpreplay.appneta.com/

[22] H. R. Loo and M. N. Marsono, "Online network traffic classification with incremental learning," *Evolving Systems,* vol. 7, pp. 129-143, 2016.

[23] G. Sun, T. Chen, Y. Su, and C. Li, "Internet Traffic Classification Based on Incremental Support Vector Machines," *Mobile Networks and Applications,* vol. 23, pp. 789-796, 2018.

[24] V. Punitha and C. Mala, "Traffic classification for connectionless services with incremental learning," *Computer Communications,* vol. 150, pp. 185-199, 2020/01/15/ 2020.

[25] G. Bovenzi *et al.*, "A First Look at Class Incremental Learning in Deep Learning Mobile Traffic Classification," presented at the Network Traffic Measurement and Analysis Conference (TMA), 2021.

[26] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, L. Ouyang, and P. Yang, "Incremental Learning for Mobile Encrypted Traffic Classification," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1-6.

[27] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Computer Networks,* vol. 207, p. 108836, 2022/04/22/ 2022.

[28] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," presented at the Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, 2009. Available: https://doi.org/10.1145/1557019.1557060

[29] V. Labayen, E. Magaña, D. Morató, and M. Izal, "Online classification of user activities using machine learning on network traffic," *Computer Networks,* vol. 181, p. 107557, 2020/11/09/ 2020.

[30] G. Xie, Q. Li, and Y. Jiang, "Self-attentive deep learning method for online traffic classification and its interpretability," *Computer Networks,* vol. 196, p. 108267, 2021/09/04/ 2021.

[31] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," in *Machine Learning and Knowledge Discovery in Databases*, Berlin, Heidelberg, 2009, pp. 254-269: Springer Berlin Heidelberg.

[32] G. Szabó, J. Szüle, Z. Turányi, and G. Pongrácz, "Multi-level Machine Learning Traffic Classification System," presented at the ICN 2012: The Eleventh International Conference on Networks, 2012.