Intrusion Detection Framework for Invasive FPV Drones Using Video Streaming Characteristics

Alsoliman, Anas; Rigoni, Giulio; Callegaro, Davide; Levorato, Marco; Pinotti, Cristina M.; Conti, Mauro

DOI
10.1145/3579999

Publication date
2023

Document Version
Final published version

Published in
ACM Transactions on Cyber-Physical Systems

Citation (APA)
Alsoliman, A., Rigoni, G., Callegaro, D., Levorato, M., Pinotti, C. M., & Conti, M. (2023). Intrusion Detection Framework for Invasive FPV Drones Using Video Streaming Characteristics. *ACM Transactions on Cyber-Physical Systems*, *7*(2), Article 12. https://doi.org/10.1145/3579999

Intrusion Detection Framework for Invasive FPV Drones Using Video Streaming Characteristics

ANAS ALSOLIMAN, University of California Irvine, United States
GIULIO RIGONI, University of Florence, Italy
DAVIDE CALLEGARO and MARCO LEVORATO, University of California Irvine, United States
CRISTINA M. PINOTTI, University of Perugia, Italy
MAURO CONTI, University of Padua, Italy - Delft University of Technology, Netherlands

Cheap **commercial off-the-shelf (COTS) First-Person View (FPV)** drones have become widely available for consumers in recent years. Unfortunately, they also provide low-cost attack opportunities to malicious users. Thus, effective methods to detect the presence of unknown and non-cooperating drones within a restricted area are highly demanded. Approaches based on detection of drones based on emitted video stream have been proposed, but were not yet shown to work against other similar benign traffic, such as that generated by wireless security cameras. Most importantly, these approaches were not studied in the context of detecting new unprofiled drone types. In this work, we propose a novel drone detection framework, which leverages specific patterns in video traffic transmitted by drones. The patterns consist of repetitive synchronization packets (we call pivots), which we use as features for a machine learning classifier. We show that our framework can achieve up to 99% in detection accuracy over an encrypted WiFi channel using only 170 packets originated from the drone within 820ms time period. Our framework is able to identify drone transmissions even among very similar WiFi transmissions (such as video streams originated from security cameras) as well as in noisy scenarios with background traffic. Furthermore, the design of our pivot features enables the classifier to detect unprofiled drones in which the classifier has never trained on and is refined using a novel feature selection strategy that selects the features that have the discriminative power of detecting new unprofiled drones.

CCS Concepts: • **Security and privacy** → *Mobile and wireless security*; **Intrusion detection systems**; • **Computing methodologies** → **Feature selection;**

Additional Key Words and Phrases: Drone detection, drone video streaming, experimental testbed

ACM Reference format:
Anas Alsoliman, Giulio Rigoni, Davide Callegaro, Marco Levorato, Cristina M. Pinotti, and Mauro Conti. 2023. Intrusion Detection Framework for Invasive FPV Drones Using Video Streaming Characteristics. *ACM Trans. Cyber-Phys. Syst.* 7, 2, Article 12 (April 2023), 29 pages.
https://doi.org/10.1145/3579999

This work was partially supported by *NSF under Grant IIS-1724331 and MLWiNS-2003237.*
Authors' addresses: A. Alsoliman, D. Callegaro, and M. Levorato, Department of Computer Science, Donald Bren Hall, 6210, Irvine, CA 92697, University of California, Irvine; emails: aalsolim@uci.edu, dcallega@uci.edu, levorato@uci.edu; G. Rigoni, University of Florence, Mathematics and Computer Science, Viale Giovanni Battista Morgagni, 67/a, 50134 Firenze FI, Italy; email: giulio.rigoni@unifi.it; C. M. Pinotti, University of Perugia, Department of Computer Science and Mathematics, Via Vanvitelli, 1, 06123 Perugia, Italy; email: cristina.pinotti@unipg.it; M. Conti, University of Padua, Department of Mathematics, Via Trieste, 63, 35131 Padova PD, Italy and Delft University of Technology, Department of Electrical Engineering, Mathematics and Computer Science, Van Mourik Broekmanweg 6, 2628 XE Delft, Netherlands; email: conti@math.unipd.it.

2378-962X/2023/04-ART12 $15.00
https://doi.org/10.1145/3579999

1 INTRODUCTION

The widespread availability of commercial drones (also known as **Unmanned Aerial Vehicles**, or **UAV** for short) and their use in a myriad of applications, pose serious security and privacy concerns. For instance, unauthorized drones may operate in restricted or crowded areas such as airports and stadiums, where they can collide with airplanes or land/crash on people. Moreover, most **Commercial-off-the-Shelf (COTS)** drones are equipped with cameras to enable a **First-Person View (FPV)**, which allows real-time video transmission from the drone's camera back to the controller (or any separate viewing device). Intuitively, this capability can easily lead to privacy violations. In response of such issues, the **Federal Aviation Administration** (**FAA** – the civil aviation regulatory body of the U.S. government) has released the final set of policies on the 15th of January, 2021, that governs the identification requirements for drones which is known as Remote ID [16]. These policies require any drone that operates in the **National Air Space (NAS)** to continuously broadcast identifying information such as a unique identifier, a timestamp, drone velocity, latitude, longitude, and altitude of both the drone and its controller. The Remote ID compliance date for drone manufacturers is September 16, 2022 and for drone pilots is September 16, 2023. However, it is difficult to enforce the compliance of such policies on the myriad of commercial drones that are pushed to the market everyday and therefore a detection solution of non-compliant drones is still needed. Many drone detection solutions exist [14, 28], but mostly remain prohibitively expensive for the majority of citizens and some of the solutions target only specific drone models [14]. These issues motivated a surge of research efforts whose objective is to provide cost-effective solutions for detecting unauthorized drones entering restricted airspace or private areas. Toward this goal, many recent contributions (e.g., [9, 21]) attempt to detect drones by analyzing their network traffic patterns.

The majority of COTS drones employ WiFi chips that allow the drone to act as a WiFi **Access Point (AP)**. The user can install an app in a smartphone/tablet to connect to the AP and establish a bidirectional data stream to receive the live-streaming FPV video and to control the drone. Most prior work on drone detection uses the FPV video stream and analyzes traffic patterns from a statistical perspective. For instance, in [21] the authors presented a framework to detect drones' FPV streams based on channel use. That approach proved challenging as video bit rates can widely vary in response to the changing scenery that is captured by the drone's camera which triggers the video compression algorithm to add/remove video frames from the stream [12, 21]. The authors in [7] use the **Received Signal Strength Indicator (RSSI)** as a feature to discern moving vs. stationary devices. However, this approach may fail to differentiate drones from other moving radio sources. In [9], the authors presented a framework that use features extracted from WiFi frames exchanged between the drone and its controllers. Similar to the previously mentioned paper, the approach is not tested with changing bit rates emitted over the FPV channel and may be unable to differentiate drones' FPV streams from other WiFi video streams such as IoT cameras [12]. Furthermore, the framework requires both the drone and its controller to be within the detection range. In [4], the authors trained a machine learning model to detect drones with high accuracy using a training set composed of drone and controller traffic. Despite using controller traffic in the dataset, the authors left the problem of the "recognition of new UAV types" and "modified video patterns" as open problems.

In this work, we propose a novel and cost-effective drone detection framework (using a WiFi adapter and a laptop/desktop) that solves the problem of detecting new drone types among other similar streaming devices based solely on the drone's FPV highly-dynamic video stream. The framework accomplishes the detection task by leveraging specific packets we identified in the encrypted FPV stream sent by drones over the WiFi channel. We refer to these specific packets, which appear in video streams at periodic intervals for synchronization purposes, as "pivots".

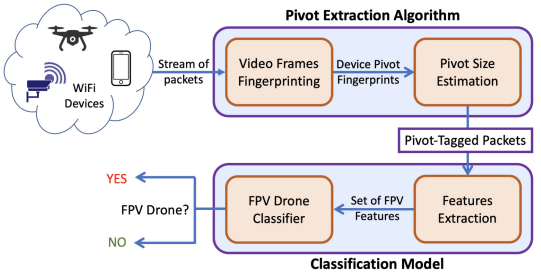


Fig. 1. Components of the drone detection framework.

The new framework consists of two components: the Pivot Extraction Algorithm and the Classification Model (Figure 1). The Pivot Extraction Algorithm identifies the pivot packets for each encountered WiFi device based on FPV video traffic patterns, while the Classification Model extracts machine learning features based on the pivot packets and trains a drone classifier.

The framework produces high drone detection accuracy under challenging settings, including (i) the scene being captured by the drone's camera is highly dynamic which produces varying bit-rates, (ii) only the drone is within the packet capture range while its controller is out of capture range, (iii) the network traffic of the drone is completely encrypted, (iv) there are other IoT devices in the environment actively emitting video streams that exhibit an FPV packet pattern similar to those generated by drones, and (v) the classifier encounters and detects drones that were not used during the training phase of the classifier.

1.1 Contributions

Under the aforementioned challenges, we summarize our contributions as follows:

- We introduce the concept of *Pivot*; a special packet found in video streams (which is used as an anchor for designing drone features) that is not affected by video parameters such as the bitrate. Furthermore, we introduce the notion of *Pivot Fingerprint*; a series of packets that most likely contains a pivot packet, and an approach for locating such fingerprints within a stream of packets.
- We design a novel feature selection strategy that is executed in two phases. The first phase is *model-independent* and based on the Jaccard Similarity Index (also known as **Intersection over Union**, or **IoU** for short) which selects the drone features that have high discriminative power to detect new unprofiled drones regardless of the machine learning model used or any unique drone behaviors that exist in a training set. The second phase is *model-dependent* and based on the **Recursive Feature Elimination (RFE)** algorithm that evaluates the sensitivity of the remaining features toward detecting unprofiled drones and selects the set of features that work best for the underlying machine learning model used in the detection framework.
- We provide an implementation and evaluation of the detection framework based on a Random Forest algorithm using features created from pivot packets that can detect drones with accuracy up to 99% even when the drones coexist with other video streaming devices such as IoT cameras and VoIP applications. We also compare our pivot features with the state-of-the-art machine learning-based drone detection features proposed by [4].

The rest of the article is structured as follows: in Section 2 we highlight the related work on the field of drone detection. Section 3 will go over a brief overview on the main components of a general drone defense system and outline how a drone detection framework fits within an inclusive drone detection system. In Section 4, we discuss the attacker and defender model, then

we provide a brief background on the characteristics of the uplink and downlink channels between the drone and its controller. In Section 5 we introduce our drone detection framework and its two components, then we discuss in details the characteristics of the pivot packets and the design of the first component; the Pivot Extraction Algorithm. We also assess the algorithm's behavior when encountered with benign video traffic such as IoT cameras. Section 6 shifts the discussion towards the framework's second components; the Classification Model, where we describe our novel pivot features creation and selection strategies, as well as our FPV drone classifier. In Section 7, we outline the implementation design of the framework in terms of hardware and software, then we describe our data collection, processing, and sampling procedures. The evaluation and results analysis of the proposed framework is reported in Section 8, while Section 9 concludes the article.

2 RELATED WORK

Drones detection is a research topic which is capturing considerable attention from the research community. Mainly, drone detection techniques fall under four different categories: (i) Radar-Based Detection [27]: based on active radars that send electromagnetic pulses toward the area to be monitored to detect the electromagnetic energy reflected by any flying objects. (ii) Vision-Based Detection: based on computer vision devices (such as regular or Thermal cameras) to detect flying objects [17, 23]. (iii) Acoustic Detection: that detects the high-pitched sound frequency generated by the rotating propellers of the drone [10, 11]. Lastly (iv) RF/WiFi Detection techniques that monitor the wireless communication links between the drone and its controller (the **Ground Control Station**, or **GCS**) [21, 22]. Since our article proposes a new WiFi detection technique, we focus on related works on RF/WiFi detection of drones. We also summarize some previous results on detection of WiFi cameras due to the similarities between the streams transmitted by the drones and those transmitted by fixed WiFi cameras.

Detection of Drone Traffic. The authors of [21] propose to detect drones by their FPV streams. The bit rate of an FPV stream emitted by a drone is compared with the bit rate of a well known and previously recorded FPV data stream. However, the more the scene changes, the higher the bit rate is, and a drone recording a highly dynamic scene might not always match a specific set of FPV bit rates. The authors also argue that since the drones move, the RSSI of their FPV channels is different from those of other WiFi video streaming services. However, they did not take into account video streaming devices that might be moving such as VoIP applications on smartphones.

In [9], drones are detected by monitoring their FPV stream sent over WiFi and applying Machine Learning models. They extracted features from WiFi frames exchanged between the drone and its controllers. As in [21], the authors do not consider either the problem of detecting drone FPV streams with changing bit rates or that of differentiating between drone FPV streams and other WiFi video streams. The same authors, in [8], took the previous work a step further by improving the robustness of the algorithm. Precisely, they aim to detect a drone flying in stealth mode (i.e., a drone that is not transmitting video).

In [19], a framework is proposed to fingerprint drone WiFi communications for the identification of specific drone models. Firstly, drones are discerned by other devices via their speed (exploiting the signal strength information used to calculate the acceleration). Once a WiFi session is associated with a drone, it is further classified using different features (i.e., pattern of probe messages and information in a Frame Header). However, this detection is based on the assumption that other devices cannot move at the same speed as the drone. In some scenarios, a device could be inside a vehicle (i.e., a phone inside a car) that moves at comparable or higher speeds than that of a drone.

In [24, 25], standard classification algorithms are used on eavesdropped traffic exchanged between a drone and its remote controller to analyze extracted features such as packets' inter-arrival times and sizes. The main aim is to detect a drone and its status, i.e., flying vs. resting.

In [4], Alipour-Fanid et al. use classical features such as those used in [24, 25] for differentiating FPV drones from other devices, and hence for detecting drones. The authors show that their framework detects with high accuracy drones using features extracted from packet samples of at least 50 packets exchanged between the drone and controller. The authors left the problems of "recognition of new UAV types" and "modified video patterns" as open problems. Our work aims to improve the results in [4] and to close the problem of recognizing new UAV types.

Detection of WiFi Cameras. In [12], the authors detect hidden wireless cameras using smartphones. Their system, DeWiCam, automatically analyzes wireless traffic to recognize camera transmissions, specifically relying on physical and MAC layer features. Importantly, camera traffic streams have relatively stable volume and packet size pattern. Besides, wireless cameras can work continuously without interruptions in the stream.

In [20], the main objective is to identify hidden WiFi cameras by altering the ambient light captured by the cameras to induce variations in the camera's packet flow (caused by the video compression algorithm) which can be identified by statistical techniques. Both papers exploit the compression mechanism that cameras use for video transmissions, discussed below in Section 4. In our work, we are also interested in investigating characteristics of video transmissions, with the different objective of detecting drones in challenging scenarios.

3 BACKGROUND ON DRONE DEFENSE SYSTEMS

A general drone defense system can be realized by deploying three independent components: drone detection, drone authentication, and drone prevention. Each component is an independent system that is designed to address a specific problem. In essence, the function of each component complements each other, but at the same time, the system design of each of them is mostly independent. For example, drone detection complements drone authentication in the sense that it would be meaningless to have a system for verifying the authenticity of flying drones without having the means of detecting the presence of unauthenticated drones in the first place, but whether the detection system is radar-based or network-based, it will not influence the accuracy of the authentication system in verifying the identity of the detected drones. Similarly, the capability of a prevention system to capture and/or destroy an unauthorized drone is not influenced by how the authorization decision was actually made but such prevention system cannot decide which drone to neutralize on its own and therefore an authorization system complements a prevention system. More specific details of each component are discussed next.

3.1 Drone Detection

The purpose of the drone detection system is the detection of the physical presence of a drone within a predefined area, typically referred to as "restricted airspace". There are different approaches for addressing the detection problem. These approaches (as previously discussed) can be categorized as radar-based, vision-based, acoustic-based, and network-based detection. The selected approach (or group of approaches) is based on the threat model. For instance, the safety concerns surrounding a popular sporting event might include the risk of having spectators using FPV drones to watch and record the match, which would result in the risk of drones crashing onto the event attendees. Based on the threat model, a network-based drone detection system might be deployed to exploit the wireless traffic generated by the FPV drone for the detection process. On the other hand, the threat model for a military installation might include risks posed by a more capable malicious adversary such as autonomous drones that do not generate any wireless traffic. Such a highly sophisticated thread model might require the deployment of multiple highly reliable drone detection systems such as radar-based and vision-based drone detection systems. Once a drone is

detected, the authentication, authorization, and neutralization/countermeasurement systems are designed and implemented based on their own threat models as well.

While the objective of the drone detection component is to detect the presence of drones, the task of tracking and pinpointing the exact physical location of the detected drone or its pilot for intervention falls under the Drone Prevention component. It is crucial to differentiate between detection and tracking since some threat models do not require a physical intervention with the invasive drones and therefore a tracking system might not be needed for the prevention component of the drone defense system. For example, AeroScope [14] is a drone detection system developed by DJI - a leading drone manufacturer - which can detect the presence of DJI drones. Every DJI drone is manufactured to continuously broadcast information about the drone's flight mission and the pilot's information. Once AeroScope detects a DJI drone via its broadcast, the AeroScope operator simply attempts to contact the drone pilot based on the information obtained from the DJI registration database. Since AeroScope is intended to be used in a civilian setting, this simple prevention strategy (contacting the pilot) might be adequate for DJI's threat model.

3.2 Drone Authentication

The authentication of drones in-flight can take different forms. For instance, it can be in the form of a broadcast message transmitted by the drone, which includes digitally signed identifying information [6]. Other forms of authentication systems propose the use of the safety flashing probes mounted on the drone frame to send visual identifying cues in the form of light pulses [5]. One of the main objectives of an authentication system is to identify the drone's identity for accountability. Another objective is to allow authorized drones to fly over a restricted airspace. For example, the FAA strictly prohibits any drone activity over national airports. However, the airspace surrounding the airport is also classified as a restricted airspace, but drone pilots are allowed to operate their drones under the exception that drone pilots obtain a flight authorization from the air traffic controller (airport control tower).

In an attempt to unify the authorization framework, the FAA and NASA have jointly developed a national identification system for drones called Remote ID [16] which enforced all drone manufacturers to equip their drones with a Remote ID module by September 16, 2022. This module continuously broadcasts identifying information regarding the drone identity and its flight status such as speed and direction. However as one can notice from such identification design is that only the complying drones can be detected by their transmitted broadcast while non-compliant drones (such as custom-made drones) are not. Therefore, a drone authentication system should be complemented by a drone detection system. Furthermore, once an unauthorized drone is detected, the authorization policy should be enforced by a drone prevention component.

3.3 Drone Prevention

Similar to the previous components, the design of a particular drone prevention system is based on the threat model. For example, Tokyo Municipal Police Office [31] proposed a guardian drone system that consists of a specialized drone equipped with a large net for capturing rogue or suspicious drones flying into restricted airspace such as near government buildings. In a more critical setting, the Israeli Ministry of Defense claimed to have designed a drone prevention system – code-named Iron Beam – which consists of a high-powered laser system that can burn attack drones in-flight with a concentrated beam of light [18]. In a more relaxed threat model such as in a civilian setting, home owners might have concerns related to a spying FPV drone. The prevention strategy in this scenario might be less invasive where a simple FPV detection system would alert the home owner to close the window blinds for example to preserve the privacy of the household members.

A notable remark about the prevention component is that if a physical intervention with drones is desired, a drone prevention system might require the drone's current physical location to acquire a lock on the target drone. Recall that (and as discussed in Section 3.1) the objective of the drone detection component is to only detect the presence of drones within the restricted airspace while tracking the detected drones for intervention is performed by the drone prevention component. This separation of duties between detection and tracking is usually desired because a drone detection system is typically designed to be always-on while a drone prevention system is only activated once a drone is detected. Therefore, it would be more cost-effective to invoke costly targeting operations only when needed. For example, consider a guardian drone equipped with a radar imaging and tracking system that can scan the sky at a narrow angle (similar to [31]). It would be impractical to continuously operate the guardian drone over the restricted airspace while monitoring all airspace angles simultaneously, given the average operating time of quad and hexacopters on a fully charged battery to be around 30 minutes. An alternative and more practical solution is to have a drone detection system in place that detects the presence of invasive drones. Then only a general direction of the drone location (for example by using multiple FPV detection stations) is provided. Therefore, only then the guardian drone can be dispatched to track and capture the invasive drone in the detected area. In a more malicious environment such as in a military setting, the threat model might require immediate and accurate location of the detected drone for a quick intervention. Therefore, the prevention system might depend on more accurate localization information by using radar-based or vision-based detection systems for example. Then based on the drone's estimated coordinates, the drone could be intercepted by a heat-seeker tracking prevention system for example.

4 DRONE'S THREAT & NETWORK MODEL

In this section, we first discuss the *attacker and defender* model and its underlying assumptions. Then we provide a brief background on the characteristics of the communication channels between the drone and its controller. Specifically, Section 4.2 describes the control channel (uplink) and its components, while Section 4.3 discusses the FPV channel (downlink) and how a video compression algorithm affects the behavior of a network traffic pattern.

4.1 Attacker and Defender Model

As discussed in Section 3, a complete drone defense system is generally comprised of three different components; (i) drone detection (detects the presence of drones), (ii) drone authentication (verifies the authenticity of drones' identities), and (iii) drone prevention (stops and/or neutralizes drones' access to the restricted airspace). Since the scope of this article is focused on drone detection, the defender model described below is formulated from a drone detection perspective only. We would like to remark that our drone detection framework is designed to be independent; can either be used as a stand-alone drone detection system or be easily integrated into other drone defense systems (e.g., authentication or prevention) without having any presumptions or specific requirements on how to operate these systems.

In our defender model (Figure 2), we consider a restricted airspace protected by a monitoring station. The station's objective is to detect unauthorized drone's access. The unauthorized drone is defined as an unknown FPV drone (in terms of type, model, or brand) that violates the access policy of the restricted airspace. This unknown drone will be denoted as "unprofiled drone" throughout this article. The monitoring station is comprised of COTS hardware; a general purpose computer such as a laptop or a desktop, and a WiFi adapter that captures all WiFi packets that are sent within its vicinity. For each stream of packets, the station analyzes the stream and labels it as either 'drone' or 'non-drone' based on drone FPV signatures. We also assume the existence of other video

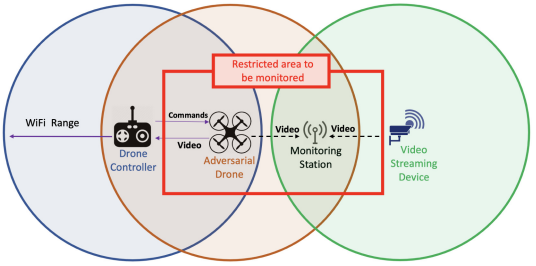


Fig. 2. Attacker and defender model.

streaming devices that can possibly generate false-alarms within the station's receiving range such as IoT cameras.

In the attacker model, we consider an unauthorized FPV drone attempting to access the restricted airspace. When the drone operates, a bi-directional connection is established between the drone and its controller. The downlink channel is a video stream sent by the drone to the controller, while the uplink channel carries control commands sent from the controller to the drone. We assume that both channels are encrypted and the drone's controller is out of the monitoring station's detection range.

4.2 Control Channel

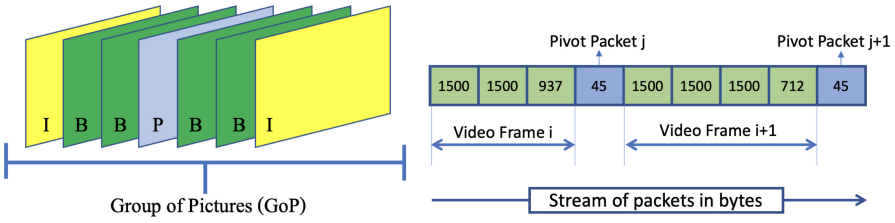
The control channel carries flight commands from the controller to the drone. Overall, the control channel comprises of two different network flows, periodic control packets and heartbeats packets. In case of controllers that receive a video stream over TCP, a flow of **acknowledgments (ACKs)** will be sent out back to the drone as well. Note that even if ACKs are encrypted, they can be easily identified by the packet size (20 bytes TCP header without options + 20 bytes IP header without payload). Some control applications embed heartbeat messages within the stream of periodic control packets. Both streams (heartbeats and control packets) have rather unique sizes and inter-arrival times. Therefore, a simple detection scheme can easily differentiate between the controller's traffic and other background traffic. In our framework, we reflect a more realistic –as well as more challenging– scenario where we assume that controllers are not always present within the range of the detection system and their traffic cannot be captured and classified (Figure 2).

4.3 FPV Channel & Video Encoding

Since the wireless channel is assumed to be encrypted, the detection framework cannot identify drone FPV channels by simply inspecting the content of the intercepted packets. Therefore, the traffic profile of an FPV transmission must be identified based on understanding how video encoding and compression algorithms affect the network traffic pattern of a video streaming application.

A drone's FPV channel is considered a channel of a video streaming application that transports compressed video frames from the drone to the controller. A video is a stream or a sequence of still pictures. In order to decrease the portion of channel capacity used to transport the video over wireless channels, almost all video encoders include a video compression algorithm.

Typically, compression algorithms exploit the – possibly high – correlation along the spatial and temporal dimensions within **Group of Pictures (GoP)**. In brief, while JPEG compression is used in the spatial domain, the core idea to harness temporal correlation is to encode differences compared to reference frames within each GoP. We, then, have three frame types: Intra-Coded Frames (I-Frames), Predicted Frames (P-Frames), and Bi-directional Frames (B-Frames). I-Frames are Intra-coded frames that exploit spatial redundancy (correlation among the pixels in the frame)



(a) GoP starts and ends with an I-Frame (b) A 45-bytes pivot packet buffered between two video frames packets

Fig. 3. A single video frame is sent over multiple packets while pivots are contained in a single packet.

to achieve compression at individual frame-level. P-Frames and B-Frames are Inter-coded frames that exploit temporal redundancy prediction. The difference between P- and B-Frames is that P-Frames only encode pixels that are changed compared to the last reference frame, while B-Frame considers both previous and future reference frames. These frames are grouped into a single GoP, which starts and ends with an I-Frame (Figure 3(a)). Typically, video streaming applications rely on well-known video codecs such as H.264 to compress raw images captured by some camera and turn them into compressed frames for faster transmission. In general, B-Frames (which includes futures changes in the frame) are used in prerecorded videos (such as YouTube) but are not used in real-time streaming applications (such as FPV video streaming) to minimize the video delay, on the expense of lowering the video quality or increasing the occupied bandwidth.

Intuitively, scene characteristics and the motion of the drone heavily influence the compression rate achieved by the scheme described above, as well as the temporal structure of the data stream. In other words, if the captured scene is dynamic, then the encoding scheme will either (*i*) generate I-Frames more frequently, which in turn results in shorter GoPs or larger P-/B-Frames, or (*ii*) generate larger differential frames. In both cases, the resulting compression gain is reduced.

The keynote of a video stream transmitted over a WiFi network is that the size of these frames is larger than a WiFi **MTU (Maximum Transmission Unit)** which is 2304 bytes. Network interfaces translate all packets sent and received from the operating system into Ethernet, which has an MTU of 1500 bytes. Therefore, each individual video frame that is larger than the MTU is fragmented into a series of packets of a size equal to the maximum Ethernet MTU size, except for the last packet, which contains the residual bytes of the frame.

FPV video streams also include other messages such as synchronization information for maintaining the state of the channel. Typically, these messages are small and fit a single WiFi packet and are sent periodically at predicted intervals. Packets from the compressed video stream are emitted more frequently compared to sync packets (the packets that contain the synchronization information), which then have a *higher probability* to be buffered right after the last less-than-MTU sized frame packet (Figure 3(b)). A key observation is that the size of these sync packets is fixed for the entire duration of the video stream.

The resulting pattern contains rather unique packet sequences. The fixed-size of the sync packets make them suitable to act as **pivots** which can be used to build FPV drone features for a detection model. The characteristics of pivot packets will be discussed in Section 5.1.

4.4 Discussion

Literature generally treats the traffic profiling of a device connected to an encrypted network, by leveraging the packet sizes and their inter-arrivals [13, 29] (i.e., without the need to look at the content of the encrypted packets). In the specific case of traffic profiles of video streaming

applications, the uplink traffic generated by the control channel is quite unique: short periodic packets (Section 4.2). Although this approach can lead to robust detectors [8, 9, 15, 24], the source of those packets is the controller, which might be out of the range of the monitoring station (Figure 2). On the other hand, the downlink traffic generated by the FPV channel can be profiled by the unique pattern of packet sizes generated by the underlying video compression algorithm (specifically, by looking at the timing and size of I- and P-Frames [12, 21]) as shown in Figure 3(b). However, the FPV channel is susceptible to the frame variations in the captured scene which would change the shape of the traffic profile for a given device depending on the scene dynamics. This change in the FPV traffic would make it difficult to build a video profile for a video streaming device, and even more difficult for FPV drones that are similar in their traffic profiles to other benign video streaming devices such as IoT cameras. In an attempt to differentiate between drone video and all other non-drone video profiles, we will discuss next the potential of using the sync packets (which we denote as pivot packets) to build machine learning features for a drone detection classifier that correctly classify drones even when other similar traffic to drone FPV exists in the detection range.

5 DRONE DETECTION FRAMEWORK

In this section we describe our *Drone Detection Framework* (Figure 1), which is based on the concept of a *pivot*; a special packet found in the drone's FPV traffic which will be utilized as an anchor for building drone features. The Drone Detection Framework is the composition of two main components: a *Pivot Extraction Algorithm* (Section 5.3) and a *Classification Model* (Section 6). In short, the Pivot Extraction Algorithm takes a sequence of encrypted packets as an input, tracks the packets that belong to video frames based on the video compression analysis described in Section 4.3, extracts pivot fingerprints, and estimates the size of the pivot packet for each encountered WiFi device. The engine of the Classification Model uses the pivot size to construct a set of FPV features and feeds it to a binary classifier which outputs "*YES*" if the packets belong to an FPV drone and "*NO*" otherwise.

This section mainly focuses on the first component of the framework, Pivot Extraction Algorithm, while the second component will be discussed later in Section 6. In this section, Section 5.1 will discuss what does a pivot packet look like via network traffic analysis. Section 5.2 describes how pivot packets appear with respect to other normal packets. In Section 5.3 we show how to locate and extract pivot packets from a network trace using our Pivot Extraction Algorithm via tracking video frames that form pivot fingerprints, while Section 5.4 illustrates how the algorithm differentiates between FPV video streams and other benign streams such as IoT cameras.

5.1 Pivot Definition via FPV Traffic Analysis

To define pivot packets, we analyze data collected from three FPV drones from three different brands: EACHINE E58 WiFi FPV Quadcopter, Spacekey DC 014 FPV WiFi Drone, and Ryze Tello Quadcopter Drone. In the following, the three drones are referred to (according to their painted color) as Black (BLK), Red (RED), and White (WHT) drone, respectively (Figure 4(a)).

Each drone has a built-in camera with a **First-Person-View (FPV)** capability. Upon powering up, the drone creates an **Access Point (AP)** and the user connects his/her smartphone/tablet to that AP. Each drone has its own app that can be downloaded from Google Play or Apple's App Store. When the user connects to the drone's AP, a video stream can be initiated and viewed on the smartphone through the drone's designated app.

For every one of the three drones, we capture decrypted network traffic (for easier initial analysis) of the drone's FPV using an external WiFi adapter set into Monitor Mode from five meters away for 30 seconds. We consider two different video states: the first state corresponds to the

(a) FPV Drones (b) IoT Cameras

Fig. 4. The FPV drones and IoT cameras used in the pivot analysis.

drone's camera capturing a **stable scene (STB)**, while the second state corresponds to a highly unstable **(shaking) drone flight (SHK)** where the camera rapidly points at different directions. The latter state induces fast changes in the captured video stream which would potentially affect the frame-rate of the video compression algorithm and ultimately the video bit-rate as explained earlier in Section 4.3.

Analyzing the BLK drone's traffic, we observe that before transmitting a series of full-sized packets (which we assume is a video frame), the drone transmits a 46-bytes packet. This packet includes an ASCII-readable command called *lewei_cmd*, which appears to be responsible for transferring media files from the drone to its associated app on the controlling smartphone [30]. The app associated with the BLK drone also sends some *lewei_cmd* packets to the drone, but only a few of them (once every second). The RED drone, instead, sends 4-bytes packets (with identical payloads) to its RED-app right before sending a video frame. The RED-app sends the same 4-bytes packet to the drone, but only four times in the entire 30 seconds network trace. Finally, the WHT drone sends an identical 35-bytes packet (except for the last 2 bytes in the payload) to its WHT-app between some video frames. These 35-bytes packets are sent at uniform intervals (every 0.1 of a second), which is independent of the varying FPV frames transmission times. For all of the three drones, traffic patterns were observed for both SHK and STB video states.

From these observations we conclude that each drone periodically sends special packets that are repetitive and identical in length, and are not part of a video frame payload. In our drone detection framework, we will leverage these packets and we name them the *pivot* packets that will be used to create features enabling drone detection.

5.2 Patterns & Behavior of Pivot Packets

We now study the occurrence and patterns of pivot packets in relation to the two different video states: SHK and STB. In both states of each drone, we compute the average bit-rate and the pivot appearance rate (pivot per second). The results are reported in Table 1. It can be observed that when the video state shifts from STB to SHK, the FPV bit-rate of the BLK drone on average increases by 444 kbps (25%), while the number of pivots observed per second increases by 2 (25%). In the RED drone, instead, the FPV bit-rate increases only by 15 kbps (0.8%), while the number of pivots observed per second increases by 2 (12%). In the WHT drone, although the FPV bit-rate increases by 334 kbps (13%), the number of observed pivots remains almost constant.

Although the collected data from the BLK drone seems to indicate a dependence of the number of pivots on the FPV bit-rate (both increased by almost the same percentage), the patterns observed in the RED and WHT drones streams instead supports the hypothesis that the pivots are not

Table 1. Pivot Frequency & Bit-rate Change

Drone	FPV Bit-Rate	Pivot/Second	Δ FPV Bit-Rate	Δ Pivot/Second
BLK-SHK	2.244 Mbps	9.2	≈ 444 kbps	≈ 2
BLK-STB	1.800 Mbps	7.6		
RED-SHK	1.817 Mbps	18.6	≈ 15 kbps	≈ 2
RED-STB	1.802 Mbps	16.6		
WHT-SHK	3.054 Mbps	8.0	≈ 334 kbps	≈ 0
WHT-STB	2.720 Mbps	8.0		

necessarily tied to the bit-rate (especially for the WHT drone since the inter-arrivals of the pivot packets are constant) and the number of pivot packets are almost the same for both the video states, that is, pivots are independent of the bit-rate increase. The deviation in the number of pivots in the other two drones might be due to (i) the nonuniform inter-arrivals of pivots and (ii) the inherent packet loss in the wireless environment for packets captured via WiFi Monitor Mode, which can affect the small number of transmitted pivots. To this end, we can safely assume that the pivots are not necessarily tied to the bit-rate (i.e., number of video frames) of the FPV stream and, as such, pivots can be used to create robust features for drone detection regardless of the video's motion state.

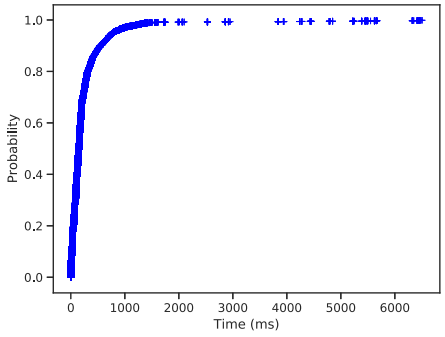
5.3 Pivot Extraction Algorithm

The Pivot Extraction Algorithm is the first component of the Drone Detection Framework, and its objective is to extract the size of the pivot packet for a given device. Assuming encrypted FPV streams, a fundamental problem is the identification the pivot packets. Herein, we take an approach based on their size. As different drones may have different formats for the pivot packets, the challenge is then to estimate the size of pivots from a network trace for each device we encounter during monitoring.

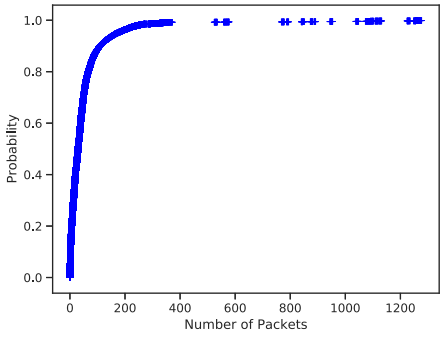
As discussed in Section 4.3, video frames are packetized by the **Network Interface Card (NIC)** and each packet's payload is capped at the **Maximum Transmission Unit (MTU)** of the transmitting NIC. Typically, the WiFi protocol has an MTU of 2304 bytes whereas Ethernet has an MTU of 1500 bytes. Since operating systems translate all sent and received packets to and from the WiFi card into Ethernet, all packets have to adhere to the Ethernet's MTU which is 1500 bytes. Therefore, video frames that are larger than 1500 bytes are packetized into a series of 1500 bytes packets where the last packet contains the residual of the video frame and its size is typically less than 1500 bytes.

Besides packetized video frames, the NIC also receives pivot packets from the drone application which, we recall, are synchronization packets sent periodically with large inter-arrivals between them (30 ms to 100 ms). On the other hand, each processed video frame is packetized and buffered at the NIC's transmission queue where each packet is sent out as soon as the wireless channel is cleared (inter-arrivals of packets belong to the same video frame ≈ 0.01 ms). Therefore, there is a high probability that pivot packets are buffered between two packetized video frames. In other words, since each packetized video frame ends with a less-than-MTU packet and begins with an MTU packet, a pivot is most likely to appear after a less-than-MTU packet and before an MTU packet (recall Figure 3(b)). From this observation, we established that a group of less-than-MTU packets between two MTU packets might include a pivot packet where the last packet is most likely the pivot. It is also observed that a pivot packet might appear after two less-than-MTU packets or slip individually between two MTUs.

During the pivot analysis, we also observed that a pivot packet has another attribute; it is sent more frequently compared to all other packet types. Based on this observation, we computed



(a) Cumulative Distribution Function (CDF) of packet inter-arrivals



(b) Cumulative Distribution Function (CDF) of packet sizes

Fig. 5. Cumulative distribution function of time and packets needed to acquire the pivot packet.

the occurrence frequency of all packet sizes within drones' traces. We observed that the highest occurring packet size is the MTU size for all of the drones we considered. Then, we observed that the second-highest occurring packet size for all three drones (BLK, RED, and WHT) is the size of the pivot packet in both motion states (STB and SHK). Knowing these characteristics, we set a new objective to determine how much time and how many packets are required to collect and have enough occurrence frequency in order to find the correct pivot packet size. To approach this objective, we calculate the distribution of the time and number of packets needed to correctly identify the size of the second highest occurring packet for every drone trace file. First, we extract the pivot size manually beforehand as the ground truth for each drone by distributing the occurrence frequency of packet sizes of the entire network trace of each drone. Then, we designate each packet in the trace as the starting point for accumulating all subsequent packets until the pivot size matches the ground truth of the trace file. Now for each starting point, we have the time and number of packets needed to correctly identify the pivot size. Finally, we calculate the **Cumulative Distribution Function (CDF)** over the detection time (Figure 5(a)) and number of packets (Figure 5(b)) for all drone traces. From the CDF, we can observe that we need at least 820 milliseconds and at least 170 packets to acquire the correct pivot size with probability of 0.95. Therefore, we set the sample size to be at least 170 packets with a time window of at least 820 millisecond.

Based on these observations, we develop a **Pivot Extraction Algorithm** that is executed in two phases, *Video Frames Fingerprinting* and *Pivot Size Estimation* (see Figure 1). During the Video Frames Fingerprinting phase, the algorithm moves a sliding window across a network traffic stream that can fit at least 170 packets sent within least a 820 millisecond window and records any packets sequence that (i) starts and ends with a packet of size exactly MTU, (ii) has fewer than six consecutive packets, (iii) and the size of all packets (except the first and last packet) is less-than-MTU. Each recorded sequence of packets are denoted as a *Pivot Fingerprint* which might include a pivot packet (Figure 6). In the *Pivot Size Estimation* phase, the recorded Pivot Fingerprints are evaluated against three types of fingerprints: type 1 contains three packets, type 2 contains four packets, and type 3 contains five packets (Figure 6). For each fingerprint, the before-the-last packet in the fingerprint is marked as a candidate pivot. Then on a separate process, the occurrence frequency of all packet sizes in the sliding window is computed. Finally, the candidate pivot size that matches the second-highest occurring packet size is declared as the size of the pivot packet.

It remains now to explain how to efficiently estimate the MTU packet size for a given sample of packets. The algorithm declares the size of the first packet in the sample as the MTU size then

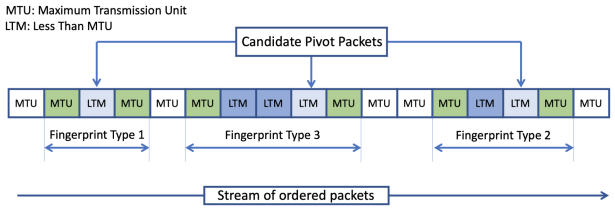


Fig. 6. Candidate pivot packets.

proceeds to find the next MTU packet. Whenever a packet size that is higher than MTU is detected, the Pivot Extraction Algorithm declares the new packet as the MTU and resumes the pivot estimation process.

Our Pivot Extraction Algorithm has correctly identified up to 99% of pivot packets within a packet trace in $O(n)$ time where n is the number of packets in the sample. Thus, the Pivot Extraction Algorithm can – with high probability – identify the pivot size.

5.4 Pivot Patterns in Other Real-time Video Streaming Devices

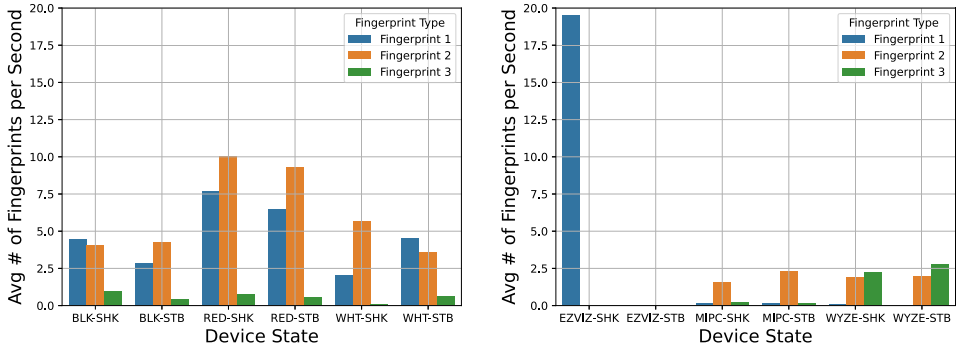
One can readily observe that the Video Frames Fingerprinting process is influenced by the video stream emitted by a device and therefore the ability of the Pivot Fingerprints in differentiating between FPV drones and other video streaming devices is questionable. We, then, analyze video streams emitted by IoT cameras in order to compare them with those emitted by FPV drones in terms of pivot patterns.

Real-time video streaming devices, such as IoT cameras, generate streams of packets that have a strong similarity with FPV drones' video streams. Analogously to FPV drones, video streams emitted by IoT cameras are unidirectional. That is, an IoT camera forwards a video stream to its designated app while the app maintains the video connectivity with the camera via heartbeats and keepalive messages. Conversely the video streams generated by VoIP applications (e.g., Skype) are bidirectional. Therefore, we incline toward using IoT cameras in the analysis of our pivot extraction approach since they are the most similar applications to FPV drones. However, we still included VoIP traces for the final evaluation in Section 7.1.2. In order to assess whether or not our pivot approach can distinguish IoT cameras from FPV drones, we collect video traces produced by three different IoT camera brands: Wyze Cam, EZVIZ C1C, and Lefun MIPC (Figure 4(b)). Just like the drones, we collect traces for each camera in a stable (STB) and dynamic motion (SHK) state, resulting in a total of six camera traces. We then calculate how many pivots are recorded per second under each pivot type, i.e., in which type of fingerprint a pivot is found.

In Figure 7, for each device (drone and camera), we compute the average number of pivot fingerprint types found per second. From the results, we can see that fingerprint type 3 rarely appears in drone videos. On the other hand, each drone has at least four type 2 fingerprints per second. In Wyze Cam and Lefun MIPC cameras, we could detect some fingerprints of type 2 and 3 ($\approx$ 2.5 pivots), but almost no fingerprints of type 1 are detected in their packet traces. Instead, the EZVIZ camera in SHK state has on average about 20 fingerprints of type 1, but almost no fingerprints of type 2 and 3 in its packet trace. From the results, we conclude that other network devices that have traffic patterns similar to drones would have different pivot patterns. Thus, a pivot-based approach can discriminate the two classes of applications.

5.5 Discussion

Pivot packets show a potential in differentiating between FPV drone traffic and other video traffic. Since the network traffic is assumed to be encrypted, pivot packets cannot be identified among the



(a) Drones Fingerprints. (b) Cameras Fingerprints.

Fig. 7. Average number of Fingerprint types per second for Drones (7(a)) and Cameras (7(b)).

stream of the intercepted packets by simply looking at the packets payload. However, the size of the pivot packet is fixed for each video stream (e.g., the size of the pivot packet for the RED drone is always 4 bytes). As discussed in Section 4.3, the video compression algorithm influences the size of non-pivot packets (packets that carry the captured video frames) and therefore it continuously alters the traffic profile. Based on the video traffic profiles, three packet patterns (denoted as pivot fingerprints which are visualized in Figure 6) are identified. These fingerprints are expected to contain the pivot packet, and the size of this packet is estimated using the Pivot Extraction Algorithm. The accuracy of the algorithm is 95% when at least 170 are captured from a given device within at least 820ms.

6 CLASSIFICATION MODEL & FEATURES DESIGN

In this section, we shift our discussion to the second component of the framework, the *Classification Model* (recall Figure 1). The classification model we adopt for the FPV Drone Classifier of our framework is the **Random Forest (RF)** algorithm. This is motivated by its low complexity and good performance in many settings. Moreover, RF is widely used in the literature to address similar problems.

We combine the RF algorithm with Grid Search; a technique where different hyperparameters are tested for a fine-tuning process. The hyperparameters that we used are: the maximum depth of the tree, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, and the number of trees in the forest. The basic idea is that the RF is repeated multiple times, with different hyperparameters, and the combinations of those hyperparameters that give the best results in the validation set inside the Grid Search, are used for the final test using the test set.

For the rest of this section, we discuss the process of utilizing pivot packets as an anchor for creating machine learning features for detecting unprofiled drones. Recall that one of the main contributions of this work is detecting drones that the FPV Drone Classifier has never trained on. In Section 6.1, we provide the list of features used by the classifier. Then in Section 6.2, we introduce two novel feature selection strategy that pick the best features for unprofiled drone detection in two phases. The first phase is based on Jaccard Similarity Index and is independent of the underlying classification model that is used in the framework, while the second phase is based on the **Recursive Feature Elimination (RFE)** technique and it involves the classification model during the selection process, meaning that its performance depends on the type of the machine learning algorithm used.

Table 2. Pivot Features

Pivot Features	Description
is_large_MTU	1 if MTU is greater than 1400 bytes and 0 otherwise
fingerprint_1	Number of fingerprint type 1 in the sample
fingerprint_2	Number of fingerprint type 2 in the sample
window_time	Total packets inter-arrivals of the sample
total_packets	Total number of packets in the sample
total_size	Sum of packet sizes in the sample
pivot_size/MTU_size	The ratio of the pivot size to the MTU size
MTU_size/total_length	The ratio of MTU size to the sample size
pivot_size/total_length	The ratio of pivot size to the sample size
MTU_count	Total number of MTU packets in the sample
pivot_count	Total number of pivot packets in the sample
MTU_count/total_packets	Ratio of MTU count to packet count in the sample
pivot_count/total_packets	Ratio of pivot count to packet count in the sample

6.1 Feature Creation

As discussed in Section 5, pivot packets have the potential to differentiate between FPV drones and other network devices, including video streaming devices that exhibit similar traffic patterns to FPV drones such as IoT cameras. This differentiating potential creates an opportunity for constructing machine learning features that have the discriminative power to classify network devices into either drones or non-drones based only on the traffic observed from a WiFi network. In particular, the features are expected to correctly classify (with high probability) drone devices among all other non-drone devices even if the machine learning model does not have a prior profile of the drone being classified. The set of features are organized into the following three groups:

6.1.1 Pivot Features. Pivot features are features composed from the output of the Pivot Extraction Algorithm. Recall that the algorithm outputs a sequence of packets that contains at least 170 packets which are sent within at least 820ms time window. Within this sequence, the algorithm also tags the pivot fingerprints; the location of pivot packets with respect to the MTU and less-than-MTU video frame packets that envelop the estimated pivots (see Figure 6).

The set of pivot features and their definitions are reported in Table 2 and is comprised of 13 features. A brief description of two pivot features is provided next:

MTU Size Feature: During the drone's pivot fingerprint analysis in Section 5.1, we observed that FPV drones tend to fill their MTU packets to the maximum (1500 bytes in RED and BLK) or near maximum (1454 bytes in WHT) payload allowable by Ethernet protocol. From this observation, we created *is_MTU_Large* feature which is set to 1 if the Pivot Extraction Algorithm declares a device's MTU is more than some threshold – heuristically set to 1400 bytes – and 0 otherwise.

Pivot Fingerprint Features: We plotted the number of pivot fingerprint type 1 and 2 in a scatter plot for FPV drones and IoT cameras in Figure 8. From the plot, the difference in fingerprint appearance behavior between drones and cameras can be observed and therefore it has the potential of having the number of fingerprints as a feature for each device. Note that the scatter plot is derived from Figure 7. Because of the insignificant number of appearances of fingerprint type 3, it has not been considered as a feature.

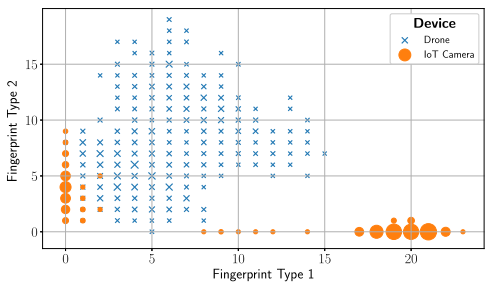


Fig. 8. Scatter plot of fingerprint type 1 & 2 for drones and cameras.

Table 3. Statistical Measurements for Computing Statistical Features

Measurements	Description
Standard Deviation	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - mean(x))^2}$
Variance	$\sigma = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - mean(x))^2$
Root Mean Square	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i)^2}$
Mean Square	$\frac{1}{N}\sum_{i=1}^{N}(x_i)^2$
Pearson Skewness	$3(mean(x) - median(x))/\sigma$
Kurtosys	$\frac{1}{N}\sum_{i=1}^{N}((x_i - mean(x))/\sigma)^4$
Skewness	$\frac{1}{N}\sum_{i=1}^{N}((x_i - mean(x))/\sigma)^3$
Minimum	$(Min(x_i)\|_{i=1,\ldots,N})$
Maximum	$(Max(x_i)\|_{i=1,\ldots,N})$
Mean	$\frac{1}{N}\sum_{i=1}^{N}(x_i)$
Median	$\lceil\frac{N+1}{2}\rceil\|(Sort_{i=1,\ldots,N})$
Mean Absolute Deviation	$\frac{1}{N}\sum_{i=1}^{N}(\|x_i - mean(x)\|)$
Median Absolute Deviation	$median(\|x_i - median(x)\|)$

6.1.2 Statistical Features. The State-of-the-Art in machine learning-based drone detection framework that exploits wireless network traffic for drone detection [4] uses 12 statistical measurements for feature generation. These features will be denoted as *statistical features* and their statistical measurements are reported in Table 3 with the addition of the "Variance" feature which was not used by [4]. Just like our framework, the drone detection framework proposed by Alipour-Fanid et al. [4] uses only packet sizes and their inter-arrival measurements for feature generation. Since both sets of features (pivot and statistical) are based on the same packet measurements, we extend our pivot-based feature set and include these statistical features to our own set as well. Including such features to our feature set is a crucial step toward performing a comparative analysis between our pivot features and the statistical features. The analysis will be conducted in Section 8.1 and it will provide an insight on the performance of our pivot features compared to the latest drone detection features proposed in the literature.

Recall that our framework assumes an encrypted network traffic and the Pivot Extraction Algorithm only extracts the size of WiFi packets and their inter-arrivals. Therefore, each statistical measurement is computed twice, once over the *packet sizes* and another over their *inter-arrivals* which results in a total of 26 statistical features (13 statistical features $\times$ 2 different packet measurements).

6.1.3 Statistical Pivot Features. The statistical pivot features combine the statistical measurements with our pivot measurements to create *Statistical Pivot Features*. From every sequence of packets received from the Pivot Extraction Algorithm, we extract three additional measurements, the inter-arrival time between pivot packets (pivot time-distance), the number of normal packets between pivot packets (pivot packet-distance), and the total number of bytes between pivot packets (pivot length-distance). For each one of the three values, we compute the statistical measurements on each group which resulted in 39 statistical pivot features (3 different pivot measurements $\times$ 13 statistical measurements). Combined with the rest of the two previously discussed feature subsets, the total feature set considered for our framework is 78 features.

6.2 Feature Selection

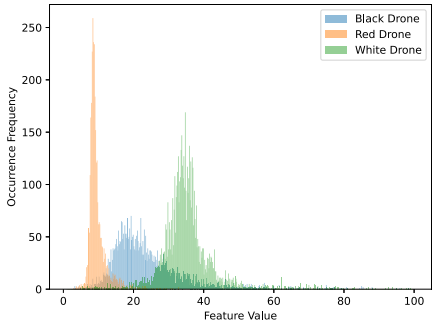
In a classification problem, conventional supervised feature selection techniques such as **Recursive Feature Elimination (RFE)** and Permutation Feature Importance aim at selecting features that better distinguish a class label for every class that exist in a dataset.

In the context of unprofiled drone detection, the feature selection process would select the features that work best in detecting profiled drones; the drones that were used in the training phase of the machine learning algorithm. In the testing phase however, the algorithm would correctly detect profiled drones while failing to detect unprofiled drones that were never seen by the algorithm during the training phase. This problem is known as overfitting, and it occurs –in the context of unprofiled drone detection– because the samples that are extracted from the same drone are correlated. To illustrate, consider two samples that are extracted from the same drone, sample SP_A and SP_B. Then consider some feature x. If we compare the value of feature x between samples SP_A and SP_B, we find that the value of x_A from sample SP_A will be statistically similar –with high probability– to the value of x_B from sample SP_B (i.e., $SP_{A_x} \approx SP_{B_x}$). In other words, because the feature values of different samples extracted from the same device are statistically similar, these samples are correlated. Consequently, if correlated samples (samples that are extracted from the same device) appear in the training set and in the testing set, it would create an undesired strong correlation between certain features and a specific device, which in turn would cause overfitting. It is a natural behavior for data points to be statistically similar to each other if extracted from the same drone because the network traffic generated by the drone is fairly unique for that drone. For example, the pivot size of WHT drone is 35-bytes. Therefore, 95% of samples extracted from WHT drone would have the "Pivot Size" feature equal to 35 (Recall that the Pivot Extraction Algorithm can find the correct pivot size with a probability of 0.95). This would cause the classifier to declare any WiFi device that has a pivot size of 46 bytes (BLK), 4 bytes (RED), or 35 bytes (WHT) as *drone* and any other pivot sizes as *non-drone* devices. This is an unwanted behavior because the model associates the unique pivot dimension with a class label, ignoring the other features. This problem will persist regardless of the dataset size; the number of packets collected from drones since all drone-labeled samples are representing (in our case) three distinct drones and their three pivot sizes are uniquely consistent with every drone sample.

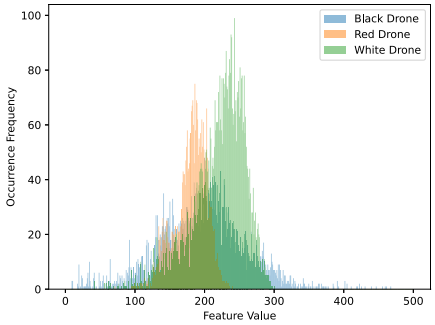
Of course this would not be a problem if we implement our model under the assumption that every drone type in the market must be profiled prior to the detection process. However, this is not a realistic assumption for creating a practical drone detection framework. To the best of our knowledge, no prior efforts that adopt a machine learning-based approach for drone detection has addressed this issue.

To this end, we next propose our novel feature selection technique that overcomes the overfitting problem via a two-phase process. The first phase computes a Feature Similarity Score based on Jaccard Similarity Index (model-independent feature selection), while the second phase applies a modified RFE for unprofiled drones (model-dependent feature selection). The objective of our feature selection technique is to select features which describe a generalized drone behavior rather than features that describe a behavior of a specific networked device. In other words, the selected features are expected to detect unprofiled drones; drones that the framework has never seen before and the classifier has never trained on.

The main motivation of applying **Jaccard Similarity Index (JSI)** testing before using an RFE-based feature selection, is that the computation of JSI scores is much faster than running an RFE-based algorithm which helps us weed out the weak features before passing them to the RFE process. For a comparison, it took our computer (a laptop with Intel i7) around 20 seconds to calculate the JSI scores for 78 features while it took the same computer around four hours to compute the RFE scores of the remaining features selected by JSI (which is 37 as we will see next). For the second phase, we opt for using a feature elimination-based technique (such as RFE) rather than a feature extraction-based technique because feature extraction techniques such as **Principal Component Analysis (PCA)** would compress and retain the overfitting features into lower dimensions rather than discarding them.



(a) Distribution of *pivot-packet-distance-mean-root-square* feature.



(b) Distribution of *pivot-packet-distance-total-size* feature.

Fig. 9. Feature (a) uniquely describes each drone individually while feature (b) generalizes an FPV drone behavior.

6.2.1 Phase I – Feature Similarity Score. As we have seen in Section 6.2, some features might describe a drone as a unique device rather than an FPV drone. For this reason, we need to identify the features that capture a generalized drone behavior and not a specific drone type. Our first step toward accomplishing this task is computing a Feature Similarity Score; a score that describes how a certain feature can capture the behavior of all the three drones we have in our experiment. Note that this feature selection approach is independent from the classification model used in the framework. In other words, Feature Similarity Score pre-processes the features without involving the adopted machine learning algorithm and therefore Feature Similarity Score results are always consistent regardless of the used classification algorithm.

We first start by plotting the distribution of each feature as a histogram across all drones and measure the similarity of their distributions. For example, Figure 9(a) represents the distributions of *pivot_packet_distance_mean_root_square* feature for all three drones while Figure 9(b) represents the distribution for *pivot_packet_distance_total_size* feature for the same drones. From the Figure we can see that the first feature represents each drone individually while the second feature shows some shared characteristics among the drones. The main intuition is that if a feature shares similar distributions among different drones then this feature will most likely generalize a drone behavior while other features with distinctive distributions can help identify the specific device it represents. An extremely distinctive distribution such as the pivot size could overfit the data.

There are different techniques in the literature for testing the similarity between data distributions such as Pearson's chi-squared test and Kolmogorov–Smirnov test. We chose the Jaccard Similarity Index method because it is easier to interpret and provides greater flexibility in setting the appropriate similarity threshold that fits our desired application.

Pre-processing: Before testing the similarity of a feature between drones, we need to prepare the data points of that feature to be plotted as a histogram. First we remove the anomalies of each feature; the data points that are scattered (very few) and are either extremely large or extremely small compared to the majority of the data points. Since we don't have a prohibitively large number of features, we removed the anomalies for each feature manually by setting the histogram range for each feature to be within the range that capture most of the data points while leaving out the very few data points that are extremely out of range. Then we perform 'data binning'; a process of dividing the range into same-length intervals, each interval (a histogram bar) is referred to as a 'bin'. A data point that falls within the range of a certain bin will increase the value of that bin by one. If anomalies are not removed, it would collapse the bins together and severely change the

shape of the histogram's distribution since the histogram range will be proportionally large for the majority of the data points. After removing the anomalies, we set the number of bins to 100 for all features as a unified bin interval.

Jaccard Similarity Index Measurement: After creating the histograms, we measure the size of the intersection area between the three drones' distributions. Then, we measure the occupation ratio of the intersection area to the total distribution area using a Jaccard index. For each bin position B_i, we record the minimum bin value among the three drones bins. The intersection area is then computed as $\sum_{i=1}^{n} min(B_{i_{1\ldots m}})$ where n is the number of bins and m is the number of histograms. Note that since we have three histograms (one for each drone), then $m = 3$. Similarly, the total area of the three histogram is $\sum_{i=1}^{n} max(B_{i_{1\ldots m}})$. Finally, the Jaccard Similarity Index JSI is computed as:

$$JSI = \frac{\sum_{i=1}^{n} min(B_{i_{1\ldots m}}))}{\sum_{i=1}^{n} max(B_{i_{1\ldots m}})} \tag{1}$$

The resulted JSI gives us the ratio of how much of the intersected region occupies from the total sum of histograms areas.

Feature Selection Based On Jaccard Similarity Index: We computed the Jaccard Similarity Index for all of the 78 features based on equation 1. The results are sorted in ascended order and reported in Figure 10. Note that the Jaccard indices are ratios ($\leq$1) and in the Figure they are multiplied by 100 for clarity. From the Figure, we notice that there are high variations between the top eight features where the index difference between features ft_i and ft_{i+1} ($\Delta JSI = ft_i - ft_{i+1}$) can reach up to 13 points between two consecutive features for the top eight indices. Then the ΔJSI started to stabilize to be ≈ 1 until it reaches to the 30^{th} feature which has $JSI = 12.71$ and $\Delta JSI = 2.64$. In other words, the number of useful features started to drop by $\Delta JSI = 2.64$ points after $JSI < 12$ and therefore we select the acceptance threshold to be 12, that is, the intersection region must occupy at least 12% of the total histogram areas. With this threshold, we reduce our features set from 78 features to 30 features. Additionally, we add another threshold that accepts a feature if it has $JSI > 70$ of intersection area between any two drones. The intuition of this additional threshold is that, a certain feature might be a strong candidate for detecting FPV drones but one of the three drones we used during testing might have a unique and outlying distribution towards that particular feature and therefore would inject an anomaly in the JSI testing which would potentially lower the JSI score of rather a strong feature. Therefore, we re-computed JSI score for each feature thrice, once between each two drones. In other words, for each feature we computed JSI for drones combinations BLK-WHT, WHT-RED, and BLK-RED. Then, we set a heuristically high threshold in which we set to $JSI > 70$ and selected the features that has $JSI > 70$ in at least one of the three drones combinations. The new threshold gave us additional seven more features which resulted in a total of 37 features.

6.2.2 Phase II – Recursive Feature Elimination for Unprofiled Drones. The features selected in Section 6.2.1 were selected independently from the underlying classification model used in the detection framework. Now we need to test the sensitivity of the remaining features and further select the features that are most sensitive toward our Random Forest classifier in detecting new unseen drones. For this task, we implement a modified version of the Recursive Feature Elimination (RFE) technique that involves our FPV Drone Classifier which influences the RFE process to select features that have the discriminative power to detect unprofiled drones (Algorithm 1).

Since RFE involves the classification model in the selection process, the model would eventually influence the RFE to select the features that would cause overfitting for the reasons discussed in the beginning of Section 6.2. To overcome this issue, we modify the basic RFE technique and inject a Cross-Validation test in the step that invokes the classifier (line 9 to 13 in Algorithm 1). Since we have three drones, we apply a 3-folds cross-validation strategy, where we divide the dataset into three distinct sets or folds (line 6).

ALGORITHM 1: Modified Recursive Feature Elimination

```
1  iteration_results.initialize();
2  while feature_set.count > 1 do
3      features_scores.initialize();
4      for ft in feature_set do
5          feat_set_no_ft = feature_set - {ft};
6          drone_subsets = k_fold_cross_validation(dataset);
7          ft_scores.initialize();
8          for drone_set in drone_subsets do
9              validation_set = drone_set;
10             train_set = drone_subsets - drone_set;
11             model = RandomForest(train_set, feat_set_no_ft);
12             accuracy = model(validation_set);
13             ft_scores.add(accuracy);
14         features_scores.add(ft, ft_scores.mean());
15     feature_set.remove_weakest_feature(features_scores);
16     iteration_results.record(feature_set);
17 final_feature_set = iteration_results.best_feature_set();
```

However, the samples are not distributed equally among the folds as in a conventional cross-validation. Instead, each fold contains samples from a single drone plus non-drone samples selected randomly. Then the RFE trains on two folds and validates on the third fold. This process is repeated once for each drone/fold as the validation set. In other words, the RFE trains on two drones and validates on new unseen drone then shuffles between the drones and trains and validates again.

The accuracy for each fold is accumulated out of 300% (# of drones × 100 at line 13) then the mean of the accuracy is computed (line 14) after the cross-validation is performed on all drones. At the end of each iteration (line 15), the feature that yields the maximum accuracy when removed (i.e., weakest feature) is eliminated. Next, a new and smaller set of features and their respective scores are then recorded (line 16) and the final feature set is selected at the end of the algorithm (line 17) based on the highest detection accuracy yielded by the best feature set. Our modified RFE process has further decreased the number of features from 37 to 12. It is worth noting that our features that are built from pivot packets are among the remaining features, and out of 28 statistical features, only three of them are kept including our 'Variance' feature in the final set. In summary, the selected features are: 3 from the Pivot Features group (is_MTU_Large, fingerprint_1, and fingerprint_2), 6 from the Pivot Statistical Features group (2 Pivot Time Distance features, 1 Pivot Packet Distance feature, and 3 Pivot Length Distance features), and 3 from the Statistical Features group (2 from Alipour-Fanid et al. [4] and our 'Variance' feature). We refer to the final set of selected features as **Pivot-Based Features**.

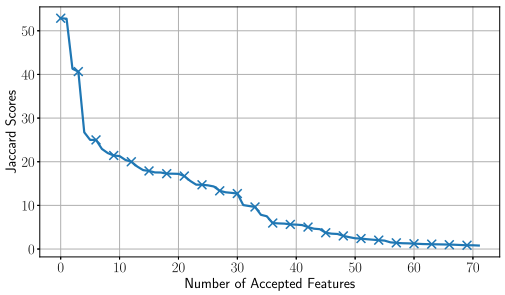


Fig. 10. Jaccard Scores × 100. Each score defines the threshold of accepted features.

7 IMPLEMENTATION

In this section, we provide an overview of our experimental setup, the devices and applications that are used to generate network traffic, and the data collection strategy used to generate our datasets.

7.1 Data Collection

This subsection provides a quick overview on the hardware and software setup used to collect the network traces to generate the framework datasets, as well as the devices and applications that generated such traces.

7.1.1 Hardware & Software Setup. To collect the traffic emitted by WiFi devices, we used an Acer Aspire F5-573G-759N laptop with Intel Dual-Core i7 CPU and Alfa AWUS036ACH WiFi adapter. We used Kali Linux 64-Bit version 2020.1, which was installed as a virtual machine using Oracle VirtualBox version 6.0.14. The WiFi adapter was set to Monitor Mode and tuned to the operating frequency of the drone's AP being monitored. We used the Android app "WiFi Analyzer" [2] to find the correct operating frequency of the AP. The network traces were collected using Wireshark [3] and parsed using Scapy [1].

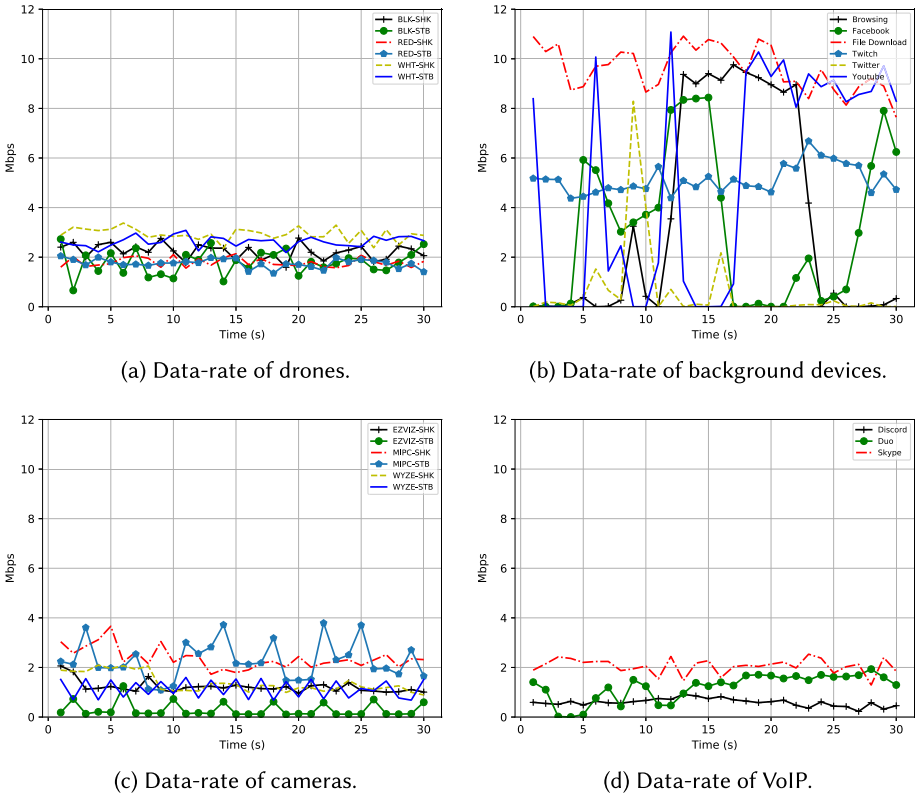
7.1.2 Network Traffic Generation. The network traffic traces are generated from various devices and applications in different conditions and scenarios. All traces are collected using the WiFi Monitor Mode (rather than on-device traffic capture) to include packet loss and other distortion effects that characterize real-world environments. The WiFi packets were captured from an encrypted WiFi network configured with a WPA2 security standard. WPA2 establishes a secure channel for each device using the CCMP protocol that uses the AES encryption algorithm to encrypt the packets payload. To maintain the state of the secure channel, CCMP attaches an 8-bytes header and an 8-bytes **Message Integrity Code (MIC)** to each packet. From each captured packet, we remove the CCMP header and its MIC which leaves us with only the WiFi header and the encrypted payload. The encrypted network traces are collected from the following devices and network services:

VoIP Apps: To test the performance of our framework, we ensure that our datasets include samples extracted from network devices that have traffic patterns similar to FPV drones'. In addition to the IoT camera traces collected for the pivot analysis in Section 5.4, we also collect traces of video traffic originated from Skype, Google Duo, and Discord VoIP apps while making video calls for five minutes. In Figure 11, we can clearly see that bit-rates of FPV drones, IoT cameras, and VoIP applications have similar traffic patterns.

Non-Drone Background Devices: To ensure the completeness of our dataset, we collect six different traces of network traffic originated from network services for three minutes each. These traces were collected during (1) file downloading, (2) non real-time video streaming (YouTube), (3) live internet video streaming (Twitch), (4) Internet browsing, and social media browsing: (5) Twitter and (6) Facebook. The bit-rate of each application is plotted in Figure 11(b). Furthermore, we included traces from publicly available WiFi packet traces [26]. Similar to all of the network traffic traces collected by us, these traces were collected in WiFi Monitor Mode which make them subject to the uncertainties of wireless networks.

Drones & IoT Cameras: The traces from drones and IoT cameras that are collected for the pivot analysis in Sections 5.2 and 5.4, respectively, are collected under two different extreme cases; a completely stable video scene (STB) and a highly dynamic video scene (SHK). To include network traffic generated from a more realistic video pattern, we place all of the three drones and all of the three cameras in front of a screen playing a five minutes documentary video that contains slow, moderate, and fast-moving scenes. From each device, we capture the emitted video traffic which resulted in an additional six network traffic traces.

Flying Drones: To further ensure that our dataset includes realistic samples, we recorded the FPV video of our drones during flight. First, we performed an experiment to determine the maximum distance from the monitoring station at which a drone can be detected. We set up a laptop as a



(a) Data-rate of drones.

(b) Data-rate of background devices.

(c) Data-rate of cameras.

(d) Data-rate of VoIP.

Fig. 11. Data-rate for drones, background devices, cameras and VoIP applications over 30 seconds.

monitoring station in the center of a university football field. Then we set up seven waypoints in a straight line starting 10 meters from the monitoring station. The distance between each two consecutive waypoints is 10 meters. We then flew each drone starting from the monitoring station in a straight line to the first waypoint while collecting the packets sent by the drone. Then, we flew the drone from the first waypoint to the second, up to the seventh while collecting the transmitted packets at each step. The drones we used are considered "micro-drones", and move at a relatively low speed with a rather small tilting of the drone body during motion.

We notice that the RSSI severely drops after 40 meters at waypoint 4 and the received power level at the monitoring station's antenna is below −80 dBm. This is expected, as these drones are low-power devices with limited battery capacity. The packet loss between waypoint 3 and waypoint 4 was excessively high (around 70%). Therefore, we conclude that in the considered hardware configuration, the effective detection distance for the micro-drones is 30 meters.

From our monitoring station in the center of the football field, we flew each drone for three minutes inside a detection area of radius equal to 30 meters while the drone occasionally exits and re-enters the area to simulate a more realistic scenario of invasive drones. For each drone, we collected the emitted video traffic which resulted in three new network traffic traces.

7.2 Data Preprocessing

For each captured WiFi packet P_i transmitted by device D_j, we extract from the WiFi packet header the transmitter's MAC address, fragment number, sequence number, header flags, payload size $S_{i,j}$,

and packet's arrival time $I_{i,j}$. We also extract the RSSI from the radiotap header that is added by the WiFi card upon receiving the packet. In case of fragmented packets, we use the sequence number, fragment number, and the "more fragments" flag in the header to put fragmented WiFi packets back into a complete packet. This approach is needed as WiFi adapters set into Monitor Mode do not reconstruct fragmented packets, and pass them individually to the operating system. We use the sequence number to identify retransmitted packets that were already captured. Note that we could not rely on the "retry" flag to capture duplicates because is not guaranteed that the original packet transmission has already been captured. Also a simple check of consecutive duplicate sequence numbers would not work either, since some retransmissions might arrive out of order, and thus the last captured packet is not always followed by its retransmitted duplicate. As an alternative solution for identifying duplicates, we simply keep track of the last eight captured packets for each D_j and whenever a new packet is received, we check if the packet is already in this window.

For each captured stream of n WiFi packets, we reconstruct the fragmented, drop the duplicates, then group them as a list of packets based on the transmitter's MAC address such as $D_j = \{P_{1,j}, \ldots, P_{n,j}\}$. For each packet i, we only save 1) the packet size S_i and 2) its arrival time I_i. Therefore, each device's packet list can be interpreted as $D_j = \{(S_{1,j}, I_{1,j}), \ldots, (S_{n,j}, I_{n,j})\}$ where $P_{i,j} = (S_{i,j}, I_{i,j})$. Then, we add each device list D_j to our Packet Dataset $DS_p = \{D_1, \ldots, D_m\}$.

7.3 Data Sampling

After preparing DS_p, the packets are grouped into batches called samples (SP). Each SP consists of at least 170 packets that are sent within at least 820ms interval (The reason for sampling packets in batches with a minimum of 170 packets and a minimum of 820ms in inter-arrival window is discussed in Section 5.3). This process transforms DS_p into a Samples Dataset DS_{sp}. To illustrate, to generate the first sample $SP_{1,j}$ for a given device $D_j \in DS_p$, a sliding-window strategy is used; the sliding-window is filled with 170 $P_{i,j}$ packets for device D_j. Then if $I_{170,j} - I_{1,j} < 820ms$, the sliding-window is filled with additional n packets until $I_{170+n,j} - I_{1,j} \geq 820ms$. Finally, the current packets in the sliding-window is recorded as $SP_{1,j}$ and added under device D_j. For the next sample, the sliding-window shifts by 30 packets to create the second sample $SP_{2,j} = \{P_{31,j}, \ldots, P_{201,j}\}$ and add additional n packets until $I_{201+n,j} - I_{31,j} \geq 820ms$ is satisfied. This sampling process continues until all packets for D_j are batched into m samples such that $D_j = \{SP_{1,j}, \ldots, SP_{m,j}\}$ where typically $n > m$ since each two consecutive samples (with at least 170 packets) have 30 interleaving packets offset. The reason for this offset choice is because during the pivot analysis, we observed that two pivot packets are separated by a maximum of 30 normal non-pivot packets. In other words, 30-packets is the maximum offset that includes all possible pivots combinations among consecutive samples.

The final dataset contains the following number of samples: Drones: {WHT Drone: 6,041 samples, RED drone: 3,194 samples, BLK drone: 4,626 samples}, Non-Drone Devices: {IoT cameras: 11,449 samples, Internet activity: 4,523 samples, VoIP apps: 1,463 samples, public traces [26]: 474 samples}. 70% of the samples of each class will be used in our RFE feature selection process (Section 6.2.2), while the remaining 30% of the samples will be retained to evaluate the performance of the model (Section 8).

7.4 Features Extraction

The last step in creating our datasets is extracting features from the samples. Since the samples are already processed, features are easily computed from each sample $SP_{i,j}$ producing a machine learning record $R_{i,j}$ that contains a set of k features Ft where $R_{i,j} = \{Ft_{i,j_1}, \ldots, Ft_{i,j_k}\}$. The feature design is already discussed in detail in Section 6.1 and reported in Tables 2 and 3 while the feature

selection strategy is discussed in Section 6.2. Since the framework follows a supervised machine learning paradigm, each record is labeled 1 if it belongs to a drone, and 0 otherwise.

7.5 Scalability

As discussed in Section 5.3, the proposed Pivot Extraction Algorithm runs linearly in $O(n)$ of time complexity. When tested on our hardware and software (Acer laptop with 2-cores i7 CPU running Python 3.8), the algorithm traversed 100,000 WiFi packets within 758ms in a single Python process. When running the algorithm through 200,000 WiFi packets using two Python processes running in parallel (each process traverses 100,000 packets), the execution time for each process was around 896ms. Processing 400,000 WiFi packets using four Python processes (each process traverses 100,000 packets), the execution time for each process was around 960ms. On average, the packet generation time of a drone is 200 packets per second. Therefore, without any code or hardware optimizations, our algorithm can process wireless traffic of 2,000 devices simultaneously using only four parallel Python processes running on a 2-cores machine (assuming every device generates 200 packets per second).

8 EVALUATION

In this section, we evaluate the detection performance of our proposed framework using the features created and selected in Section 6 and the dataset generated in Section 7. During the feature selection process, we used only 70% of the dataset and kept the remaining 30% for the final evaluation. With the remaining 30%, we employ a 3-folds cross-validation technique. In a conventional cross-validation, the dataset is split into training and testing sets by specific ratios (such as 80/20). However, to test the framework's ability in detecting unprofiled drones, we split the remaining 30% into training and testing sets by drone type. In particular, the training set is designated to contain samples of two different drones plus randomly selected non-drone samples drawn from the designated 30% pool, while the testing set contains samples from the third drone plus randomly selected non-drone samples drawn from the designated 30% pool as well. To balance the datasets, the non-drone samples are added to each set (training and testing) in a 50/50 drone to non-drone ratio. To create the second train/test set pair, we move the test set drone samples to the training set while moving one of the training set drones into the testing set. Then we redistribute the non-drone samples to 50/50 drone to non-drone ratio. The third train/test set pair follows the same process; designate the test set for the last drone then redistribute the non-drone samples. This 3-folds cross-validation setup covers all possible combinations of having two drones in the training set and the third drone in the testing set. Finally, to assert that the initial traffic analysis of the three drones used in Sections 5.1 and 6.2 did not influence or interfere with the results of the evaluation of our experiments, we collected additional samples from a fourth drone as a baseline; a 3DR Solo drone (Figure 12). The Solo drone was never used in any shape or form in neither the analysis of the pivot behavior nor the feature selection process. The samples of the new drone were collected in the same manner as in Section 7.1 and is comprised of 398 samples. Similar to the previous three train/test set pairs, we use the Solo drone to create a fourth train/test set pair where we designate the samples of the Solo drone to be part of the test set.

The evaluation of the framework is divided into two sets of experiments, each set is executed under four different experiment settings. In the first set, we apply the features that were selected by our feature selection strategy proposed in Section 6.2 which is comprised of our pivot-based features. In the second set, we apply the statistical features only (Table 3) which was proposed by [4]. Recall that one of the main motivations of the proposed framework is detecting unprofiled drones that the framework has never seen before. To test the framework's performance in detecting unprofiled drones, we designate a drone to the testing set that was never used in the

Table 4. Final Results

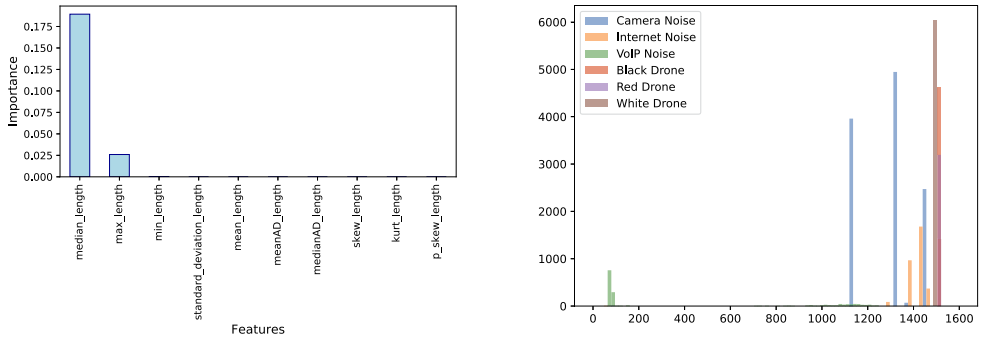
	# of samples per device in training set	# of samples per device in testing set	Pivot Features			Statistical Features		
			Accuracy	Precision	Recall	Accuracy	Precision	Recall
Experiment 1	Red Drone = 959 samples Black Drone = 1388 samples Noise Samples = 2347 samples	White Drone = 1812 samples Noise Samples = 1812 samples	97%	98%	97%	49%	0%	0%
Experiment 2	Red Drone = 959 samples White Drone = 1812 samples Noise Samples = 2771 samples	Black Drone = 1388 samples Noise Samples = 1388 samples	93%	98%	87%	69%	99%	39%
Experiment 3	White Drone = 1812 samples Black Drone = 1388 samples Noise Samples = 3200 samples	Red Drone = 959 samples Noise Samples = 959 samples	99%	98%	99%	89%	99%	79%
Experiment 4	Red Drone = 959 samples Black Drone = 1388 samples White Drone = 1812 samples Noise Samples = 4159 samples	Solo Drone = 398 samples Noise Samples = 398 samples	98%	97%	99%	50%	0%	0%

training set. In particular, the setup of the four experiments are the following; for each experiment, the classifier is (1) trained on RED and BLK and tested on WHT, (2) trained on WHT and RED and tested on BLK, (3) trained on BLK and WHT and tested on RED, and (4) trained on RED, BLK, and WHT and tested on Solo. As mentioned before, to balance the datasets (training and testing) for each experiment, we include non-drone samples in a 50/50 drone to non-drone ratio. Note that in each experiment, we construct a new Random Forest algorithm and fit it with its designated training set to make sure the new drone used in the testing set is unprofiled and never seen by the classifier. The setup and results for each experiment is reported in Table 4.

8.1 Results Analysis

The results reported in Table 4 demonstrate that pivot-based features can detect unprofiled drones with accuracy of at least 93%. On the other hand, statistical features – such as those used in state-of-the-art machine learning-based drone detection – struggle at detecting unprofiled drones, especially when detecting WHT and Solo drone; statistical features failed to detect even a single WHT or Solo drone sample. Recall that the network traffic originated by the Solo drone is completely new and unknown to the framework and were never used neither during the analysis of FPV drone behavior (Section 5.1) nor during the RFE feature selection process (Section 6.2.2). To understand the drawbacks of the statistical features, we apply a permutation feature importance technique to score the importance index for the statistical features only (Figure 13(a)). From the Figure, we observe that the one and only important feature is the median; the middle value of a sorted list of values. As discussed in Section 4.3, a video stream is comprised mostly of MTU packets and therefore the median of a series of 170 packets is almost always the MTU size; maximum packet size in the sample. The second most important feature is the packet with the maximum payload (i.e., MTU) which essentially reflects the same value of the median feature. Then we plot the distribution of the MTU for each device in Figure 13(b). As we can observe, there is almost no variance in the distribution of the MTU. In other words, every device has a unique MTU size that is tied to its network interface card (NIC) and is represented as a single bar in Figure 13(b). This explains why the classification model privileges this single feature in the detection decision. In fact, the model can associate MTUs of 1500 bytes

Fig. 12. 3DR solo drone that was used as part of the final test set.



(a) Permutation Feature Importance for Statistical Features.

(b) Distribution of the MTU feature. Note that "Camera Noise" is comprised of three distinct IoT cameras.

Fig. 13. Fixed-valued features such as MTU force the classifier to heavily rely on a single feature.

(RED and BLK drones) and 1482 bytes (WHT drone) to drone devices and any other MTU sizes to a non-drone device. This also explains why the detection accuracy drops to 55% when WHT drone is removed from the training set (since RED and BLK have the same MTU size).

On the other hand, our feature selection strategy ensures the removal of features that have a single or very few distinct values across samples drawn from the same device such as MTU or pivot size which have the potential to uniquely identify a device rather than a group of devices that share the same characteristics (i.e., FPV drones). Furthermore, the selected features are mostly influenced by the behavior of pivot packets that indeed capture a generalized drone behavior rather than a single unique device behavior.

9 CONCLUSIONS

In this article, we proposed a COTS Drone Detection Framework that can detect invasive FPV drones flying into a restricted area without the need of having a prior profile of the invasive drone or requiring the drone's controller to be within the detection range. The framework relies on synchronization packets that are interleaved with the video stream emitted by drones. These packets are denoted as "pivots" and unlike prior work, pivot packets are not affected by the varying bit-rate of the drone's video stream which can be used as a guideline to construct features for a machine learning-based system built from a Random Forest classifier. We also proposed a *Pivot Extraction Algorithm* algorithm that quickly searches a sample of packets for pivots in linear time. Furthermore, we proposed a two-phase feature selection strategy that selects drone features which have the discriminative power to detect unprofiled drones. The first phase is a model-independent feature selection process which is based on Jaccard Similarity Index, while the second phase is model-dependent and based on the RFE selection process.

Our experiments demonstrated that pivot-based features can detect unprofiled FPV drones even when other video streaming devices such as IoT cameras are within the detection environment. The detection accuracy using our pivot-based features is at least 93% using at least 170 captured packets that are transmitted within at least 820ms, while the detection accuracy of using the state-of-the-art drone features [4] is at least 49% using the same number of packets and detection window.

REFERENCES

[1] 2020-03-13. Scapy. (2020-03-13). https://scapy.net/.
[2] 2020-03-13. Wifi Analyzer. (2020-03-13). https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer.
[3] 2020-03-13. Wireshark. (2020-03-13). https://www.wireshark.org/.

[4] Alipour-Fanid et al. 2019. Machine learning-based delay-aware UAV detection and operation mode identification over encrypted Wi-Fi traffic. *IEEE Transactions on Information Forensics and Security* 15 (2019), 2346–2360.

[5] Anas Alsoliman, Marco Levorato, and A. Chen. 2021. Vision-based two-factor authentication & localization scheme for autonomous vehicles. In *Third International Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021 (part of NDSS)*.

[6] Anas Alsoliman, Abdulrahman Bin Rabiah, and Marco Levorato. 2020. Privacy-preserving authentication framework for UAS traffic management systems. In *2020 4th Cyber Security in Networking Conference (CSNet)*. IEEE, 1–8.

[7] Simon Birnbach, Richard Baker, and Ivan Martinovic. 2017. Wi-fly?: Detecting privacy invasion attacks by consumer drones. (2017).

[8] Bisio et al. 2018. Improving WiFi statistical fingerprint-based detection techniques against UAV stealth attacks. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.

[9] Bisio et al. 2018. Unauthorized amateur UAV detection based on WiFi statistical fingerprint analysis. *IEEE Communications Magazine* 56, 4 (2018), 106–111.

[10] Busset et al. 2015. Detection and tracking of drones using advanced acoustic cameras. In *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*, Vol. 9647. International Society for Optics and Photonics, 96470F.

[11] Ellen E. Case, Anne M. Zelnio, and Brian D. Rigling. 2008. Low-cost acoustic array for small UAV detection and tracking. In *2008 IEEE National Aerospace and Electronics Conference*. IEEE, 110–113.

[12] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting hidden wireless cameras via smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 1–13.

[13] Mauro Conti, Giulio Rigoni, and Flavio Toffalini. 2020. ASAINT: A spy app identification system based on network traffic. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*. 1–8.

[14] DJI. 2021-05-21. DJI AeroScope. (2021-05-21). https://www.dji.com/aeroscope.

[15] Martins Ezuma, Fatih Erden, Chethan Kumar Anjinappa, Ozgur Ozdemir, and Ismail Guvenc. 2019. Micro-UAV detection and classification from RF fingerprints using machine learning techniques. In *2019 IEEE Aerospace Conference*. IEEE, 1–13.

[16] U.S. Department of Transportation Federal Aviation Administration. 2021-03-09. UAS Remote Identification Overview. (2021-03-09). https://www.faa.gov/uas/getting_started/remote_id/.

[17] Sai Ram Ganti and Yoohwan Kim. 2016. Implementation of detection and tracking mechanism for small UAS. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 1254–1260.

[18] Isabel Kershner. 2022-06-03. Israel Builds a Laser Weapon to Zap Threats Out of the Sky. (2022-06-03). https://www.nytimes.com/2022/06/03/world/middleeast/israel-laser-rockets.html.

[19] Li et al. 2017. Drone profiling through wireless fingerprinting. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 858–863.

[20] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 243–255.

[21] Nassi et al. 2019. Drones' cryptanalysis-smashing cryptography with a flicker. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1397–1414.

[22] Phuc Nguyen et al. 2017. Matthan: Drone presence detection by identifying physical signatures in the drone's RF communication. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. 211–224.

[23] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. 2015. Flying objects detection from a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4128–4136.

[24] Sciancalepore et al. 2019. Detecting drones status via encrypted traffic analysis. In *Proceedings of the ACM Workshop on Wireless Security and Machine Learning*. 67–72.

[25] Sciancalepore et al. 2019. Picking a needle in a haystack: Detecting drones via network traffic analysis. *arXiv preprint arXiv:1901.03535* (2019).

[26] Gursimran Singh, Harish Fulara, Dheryta Jaisinghani, Mukulika Maity, Tanmoy Chakraborty, and Vinayak Naik. 2022. CRAWDAD dataset iiitd/wifiactivescanning (v. 2019-06-05). Downloaded from https://crawdad.org/iiitd/wifiactivescanning/20190605. (March 2022).

[27] SpotterRF. 2020-03-13. Drone Detection System. (2020-03-13). https://spotterrf.com/applications/drone-detection/.

[28] Robin Radar Systems. 2021-05-21. Robin Radar Systems. (2021-05-21). https://www.robinradar.com/.

[29] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2018. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2018), 63–78.

[30] Junia Valente and Alvaro A. Cardenas. 2017. Understanding security threats in consumer drones through the lens of the discovery quadcopter family. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. 31–36.

[31] Keith Wagstaff. 2015-12-11. Tokyo Police to Deploy Net-Carrying Drone to Catch Rogue Drones. (2015-12-11). https://www.nbcnews.com/tech/tech-news/tokyo-police-deploy-net-carrying-drone-catch-rogue-drones-n478596.

Received 19 July 2021; revised 23 November 2022; accepted 19 December 2022