AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-0632

Embedded Audio Processing and Neural Networks for Uncrewed Aircraft Detection

Vincent Stafford *, David Canales†, Avinash Muthu Krishnan‡, and Robert Moskowitz§
Embry-Riddle Aeronautical University, Daytona Beach, Florida, 32114
HTT Consulting LLC, Oak Park, Michigan, 48237

With the increasing prevalence of uncrewed aircraft (UAs) in industry, the need for low-cost and reliable UA detection systems has become critical for the safety and security of firms of all sizes. This article presents a novel UA detection method using transfer learning from acoustic event classification convolutional neural networks (CNNs) and distributed acoustic sensors for automatic feature extraction on embedded hardware. Testing demonstrated an average accuracy of 95.372% in UA presence detection and 99.185% in UA-type classification. This accessible and efficient model for real-time UA detection demonstrates a lower-cost, scalable alternative for rapid and distributed deployment in resource-constrained environments.

I. Introduction

In recent years, uncrewed aircraft (UAs), more commonly known as drones, have become integral to manufacturing, commerce, and military operations. However, their widespread adoption has raised important questions about their potential dangers. For instance, in June 2023, a single stray drone caused the complete shutdown of Pittsburgh International Airport, which delayed all departures and arrivals for an hour and cost the airport millions in lost revenue [1]. Furthermore, articles on using uncrewed aerial vehicles to gain information about homeowners and corporations illegally have become increasingly common in recent years [2, 3]. To respond to this, agencies such as the Federal Aviation Administration (FAA) have enacted Title 14 [4] requiring all Part 107 UA pilots to broadcast their drones' locations while in flight publicly. Although these measures help, there is still a regulatory gap for drones smaller than 250 grams [4] and for unauthorized UA use in residential and government areas by non-Part 107 UA operators. This has created a security gap that various firms and academic institutions have attempted to fill with the creation of real-time UA detection systems.

The state of the art in drone detection has varied significantly over the years but can be broken down into four types based upon their mode of detection: radar-based detection systems, radio-frequency (RF)-based systems, acoustic-based systems, and vision-based systems [5]. RF-based systems are the most proliferated currently, and in recent years, have been pioneered by DJI's Aeroscope and FAA's Remote ID. Aeroscope works for all DJI UAs and provides information, including a unique flight identification number to differentiate rounds, the UA's serial number, and flight information such as heading and altitude over a proprietary 2.4GHz interface [6]. An obvious disadvantage of this system is that it only works with UAs manufactured by DJI. In contrast, Remote ID works on similar principles but operates using short-range low-power 2.4GHz RF signals often from a dedicated transponder [6]. UAs manufactured prior to the development of the Remote ID system may not have the transponder required to be detected by Remote ID, which adds an additional cost burden to consumers and businesses alike. Furthermore, there are many attacks to spoof Remote ID communication signals or intentionally disable such transponders in UAs [7]. Radar-based, acoustic-based, and vision-based systems do not rely on specific platforms or hardware components in UAs. Rather, they attempt to pick up on a characteristic unique to UAs, such as their appearance, sound, or movement, and attempt to differentiate them from surrounding phenomena. Acoustic-based approaches largely use support vector machines and the statistics of acoustic waveforms to differentiate a UA's waveform from acoustic signals [8, 9]. The cost of acoustic-based systems is lower than other detection models but operates in much shorter ranges. Radar-based systems are the most established of the three technologies but incur the highest costs [5]. This technology evolved from the Doppler radar spectrum analysis, initially used for aircraft detection during the Cold War [10, 11]. Finally, vision-based systems arose from

*Undergraduate, Currently: Interdisciplinary Mathematical Sciences, Honors College, Florida Atlantic University, Jupiter, FL, 33458, USA.
†Assistant Professor, Department of Aerospace Engineering
‡Assistant Professor, College of Aviation
§Owner, HTT Consulting LLC

developing effective Convolutional Neural Networks (CNNs) architectures for image recognition, such as Faster R-CNN and MobileNet [12, 13].

Our proposed approach differs from previous work on UA detection by utilizing CNNs typically used in vision-based UA detection to extract features and learn from collected acoustic waveforms automatically. This model of detection stands in stark contrast to previous works, which use support vector machines (SVM) and manual feature extraction to extract meaningful information from acoustic waveforms [8, 9, 14]. This approach also introduces unique computational requirements due to the high bandwidth of data transmission required, which led to the use of transfer learning and efficient data processing techniques to permit the use of embedded devices to keep costs as low as possible. Overall, the proposed system demonstrated an accuracy of 95.372% to identify the presence of UAs, even in noisy environments, which is a significant increase from the median of 86% in authors' previous work [8] while maintaining a cost per sensor of only $18.23 USD. The structure of this paper is as follows: Section II outlines the proposed and constructed acoustic systems used for training acoustic-based neural networks and the algorithms utilized. Section III evaluates the performance of these trained neural networks. Finally, Section IV explores tentative findings and explains the results realized in Section III.

II. Methodology

The methodology contains two major developments in the project. The first focus will be on the software, algorithmic developments in the realm of data acquisition and processing. This will define the algorithms used to efficiently assimilate acoustic data into classification models and expand on the design of neural networks for UA classification. The secondary overall focus is the developments in hardware to detect UAs in the proposed model.

A. Introduction to the Acoustic UA Detection Prototype

This paper presents a model of UA detection that utilizes multiple embedded devices that stream their samples to a processing computer that detects UAs. In processing these samples, the computer utilizes a neural network to detect the presence of the acoustic signatures of UAs in the samples. If positive, the processing computer then classifies the type of drone in the sample using a second neural network. This proposed system presents a significant advantage due to reducing the unit cost of sensors since each sensor only needs to have enough computational power to send samples it gathers over the network. Despite the cost advantages of this approach, it also raises multiple algorithmic challenges; particularly the need for efficient and immediate transmission of acoustic samples from the recording devices. Then, to be able to process these samples in real time, the algorithmic implementations on the side of the computer must also be as efficient as possible. In Section II.B, the algorithms developed to pursue this research are outlined and presented, including the theoretical framework underpinning the data structures and the algorithms used for efficient data handling within the proposed UA detection model.

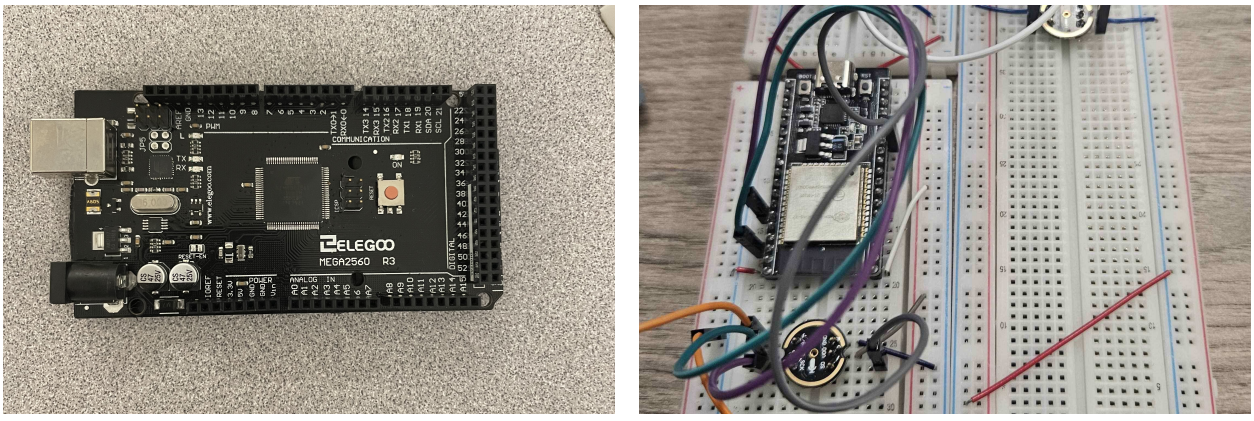

Fig. 1 Elegoo Mega 2560 microcontroller (Left) and ESP32 (Right)

Then, to evaluate this novel mode of UA detection, the authors constructed two different data collection prototypes. The first used an Arduino Mega 2560 as the microcontroller platform with electret microphones. In testing, the processing computer was represented by a Lenovo Legion 7i Generation 7 laptop running POP OS 24.04. Finally, to

evaluate the viability of the detection system in a more realistic configuration, a second system consisting of 3 sensors is constructed, using IMNP441 digital microphones and ESP32s as the microcontroller (Figure 1). Section II.C depicts the step-by-step construction of the two prototypes, their use in field testing, and the construction of the dataset used to train neural networks. These configurations form the basis for the evaluation of the performance of the networks and provide insight into the model's success in achieving accurate detection and classification.

B. Algorithmic Developments

In designing a scalable UA detection system, this paper proposes three fundamental algorithmic advances to address the computational and logistical challenges of constructing a real-time detection system using embedded devices. First, this section proposes using transfer learning from a pre-trained audio classifier to extract high-level features as embeddings from raw audio samples automatically. Using pre-trained models to extract acoustic features permits the benefits of CNNs without their high computational overhead. Next, a unified data structure is constructed to manage and package acoustic samples on resource-constrained microcontrollers. This data structure accommodates variable sampling rates and sample depths to function within the constraints of any microcontroller used and be serializable for reconstruction on the processing computer. Finally, we propose the development of an algorithm for flagging samples from different sensors at similar times. This algorithm is straightforward yet runs in just $O(m \log m)$ time, allowing for the extraction of time-synchronized samples required for applications such as multilateration. Collectively, these three algorithmic developments form a cohesive pipeline for the processing of acoustic samples from their acquisition to classification and integration into more extensive algorithmic processes.

1. Embedding Space and Transfer Learning

An embedding is a mapping of data points from an input space to a continuous lower-dimensional manifold such that the structural and semantic meanings between the points are preserved. Specifically, an embedding space is a metric space where the distance between points corresponds to their similarity based on a similarity metric, M. Points that are structurally or semantically similar to each other have smaller distances. Therefore, for any three points $x_1, x_2, x_3 \in X$ where X is the input space of the neural network N, and (E, d) are the field and distance metrics representing the embedding space, the embedding function $f : X \to E$ satisfies:

$$d(x_1, x_2) \leq d(x_1, x_3) \text{ if and only if } M(x_1, x_2) \leq M(x_1, x_3) \tag{1}$$

In the case of neural networks, the embedding function is created by passing an input vector through every layer except the final classification layer. Formally speaking, for a neural network N of L layers, the embedding function is the composition of network layers $f^{(L-1)} \circ f^{(L-2)} \circ ... \circ f^{(1)}$ and the similarity metric is represented by the Euclidean distance between the magnitudes of vectors in the embedding space.

Empirical research has demonstrated the preservation of semantic similarity in embedding spaces. For instance, the developers of AlexNet found that their neural network grouped vectors in the embedding space such that vectors with similar predictions were close to each other [15]. Transfer learning takes advantage of the notions of embedding space by extracting embeddings from pre-trained models to create new datasets. Since these embeddings are numerical vectors, one can train a dense neural network (DNN) from them. Therefore, new categories of data can be trained from the semantics and feature extraction of pre-trained models at a fraction of the computational cost. To conduct UA detection, the embeddings of YAMNET are used to encode some semantic information about the acoustic features present in the waveforms passed to it. Then, these embedding vectors are constructed into a dataset from which a simpler DNN trains. This approach balances the computational cost and benefits of automatic feature detection that CNNs introduce.

2. Complexity of Packaging and Transmitting Samples

Naturally, the process of acoustic sample transmission begins at the level of individual sensors, which work in various ways to convert the pressure differentials that represent sounds into a readable digital format. Fortunately, due to the development of unified standards of acoustic sample transmission at the embedded level, such as I^2S, one can assume that regardless of the acquisition method, all acoustic samples are received by microcontrollers as a stream of n-bit integers at a pre-specified sample rate. Depending on the size of the integers, also known as their sample depth, the amount of data collected each second can quickly overwhelm many microcontrollers. According to Table 1, on hobbyist microcontrollers such as the Arduino Mega 2560, a single second of 8-bit PCM audio sampled at 8 kHz would exceed its memory capacity. In contrast, in commercially used microcontrollers, such as the ESP32, the maximum

possible stored amount increases to about 20 seconds under the same conditions. Furthermore, since the proposed model utilizes YAMNET for transfer learning, which requires acoustic waveforms sampled at 16 kHz in a 32-bit PCM format, collecting and transmitting audio samples is significantly more resource-intensive.

Sample Rate (KHz)	Sample Depth	Required per-Second Transmission Rate
8	8-bit	8KB
	16-bit	16KB
	32-bit	32KB
16	8-bit	16KB
	16-bit	32KB
	32-bit	64KB
32	8-bit	32KB
	16-bit	64KB
	32-bit	128KB
44.1	8-bit	44.1KB
	16-bit	88.2KB
	32-bit	176.4KB

Table 1 Per-second data transmission requirements given depth and sample rate

To achieve an efficient and scalable system, a unified data structure for use across all embedded recording devices is constructed. This unified data structure allows the processing computer on the network to process acoustic samples on different platforms using identical algorithms. Furthermore, this data structure is adapted to contain a variable length of samples, allowing the scaling of data output to work within the constraints of individual hardware platforms. This adaptability is critical to bridge the gap between the high fidelity of data required by YAMNET and the feasibility of using consumer-grade microcontrollers as platforms for acoustic sensors. Figure 2 illustrates the audio frame data structure, with 8000 acoustic samples at a 16-bit sample depth, providing a visual representation of the organization required of the data. Then, to address the computational restrictions of microcontrollers, only linear time algorithms will be used when possible to serialize, transmit, and deserialize all acoustic samples. These constraints minimize the computational load of sample gathering while allowing data transmission to be scaled without significant computational overhead. These motivations lead to a formal list-based implementation of a data structure to contain the metadata and samples gathered by microcontrollers:

Definition 1. *An **audio frame** is defined as the vector:*

$$\begin{bmatrix} 1, & d, & 1, & t, & 1, & s_1, & s_2, & \cdots, & s_n, & 1 \end{bmatrix}^T \tag{2}$$

where:

- $d \in \mathbb{Z}_{256}$ *is a unique device identification number.*
- $t \in \mathbb{Z}$ *is the timestamp.*
- $s_i \in \mathbb{Z}$ *for* $i = 1, 2, \ldots, n$*, are the individual acoustic samples.*

With this definition in mind, it becomes crucial to determine the computational complexity of adding and modifying entries in the audio frame. The linear time complexity of these operations on the audio frame ensures that microcontrollers can handle many samples without introducing significant overhead.

Proposition 1. *Populating an audio frame with n samples takes* $O(n)$ *time.*

Proof. Assume $\exists \overrightarrow{v} \in \mathbb{Z}^{n+6}$ such that $v = \overrightarrow{0}$. Naturally, populating all $n + 6$ elements of $\overrightarrow{v}$ requires $n + 6$ $O(1)$ operations. Therefore, the complexity of populating $\overrightarrow{v}$ is $O(n + 6)$, which is equivalent to $O(n)$. □

Proposition 2. *Let n be the number of samples in an array, specifically within the context of the audio frame. Consider a function:*

$$f : (\mathbb{Z}^n, z, i) \to \mathbb{Z}^{n+1} \tag{3}$$

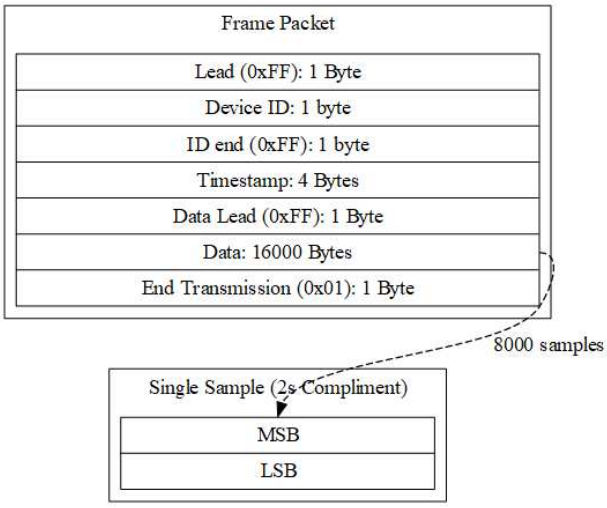

Fig. 2 Audio frame data structure with 8000 samples in 16-bit PCM

where $z, i \in \mathbb{Z}$. The function that inserts z at position i of the passed vector is $O(n)$ in the worst-case scenario.

Proof. Recall that f produces a vector $\overrightarrow{v}' \in \mathbb{R}^{n+1}$ by inserting an element $z \in \mathbb{Z}$ into position i of the vector $\overrightarrow{v} \in \mathbb{Z}^n$. Algorithmically, this function can be implemented as an iterative setting of $\overrightarrow{v}'_m = \overrightarrow{v}_m \forall m \in \{1, ..., i-1\}$, then set $\overrightarrow{v}'_i = z$. Finally, assign $\overrightarrow{v}'_{m+1} = \overrightarrow{v}_m, \forall m \in i, ..., n$. Naturally, this consists of $(i-1) + 1 + (n-i+1) = n+1$ operations. Therefore, this function takes $O(n)$ time. □

With the fundamental operations of the audio frame defined, attention must be turned to the transmission of the samples. Despite the efficiency of constructing data structures composed of integers, digital electronics do not compute or communicate in terms of abstract integers; instead, they manipulate bits and bytes that represent larger mathematical structures. The disparity between the mathematical and physical realization of computations explains the importance of serializing data structures. All complex data structures on any computer are eventually merely a string of bytes in memory. This requirement, combined with the abstract definition of the audio frame, leads to the development of a fundamental and abstract representation of what a byte is to bridge the theoretical gap:

Definition 2. *The **byte ring** is defined as the quotient ring $\mathbb{Z}/256\mathbb{Z}$.*

By Definition 2, each byte of data is an element of the commutative ring $\mathbb{Z}/256\mathbb{Z}$, which contains the equivalence classes of the modulo 256 integers. This representation provides a natural abstraction for a byte as a modular entity, where each equivalence class contains all identical integers modulo 256. For example:

$$[3] \in \mathbb{Z}/256\mathbb{Z} = \{\cdots, -253, 3, 259, 515, \cdots\}$$

Then, modules of the byte ring represent vectors of multiple bytes. For example, $\overrightarrow{v} \in (\mathbb{Z}/256\mathbb{Z})^n$ indicates a vector of length n such that each component of it is in one of the equivalence classes mentioned above. For clarity and simplicity, the square brackets denoting an equivalence class in the vector are omitted. These vectors can represent any arbitrary byte string, and thus, serialization is described as a projection from a vector over the integers to a vector over the byte ring.

Definition 3. *A vector $\overrightarrow{v} \in \mathbb{Z}^n$ is said to be **serialized** if and only if each component $v_i \in \overrightarrow{v}$ is the least non-negative integer representative of its equivalence class in $\mathbb{Z}/256\mathbb{Z}$.*

With the definition of serialization set, it is naturally required to consider what the act of serialization looks like algorithmically. To do so, one must first prove that converting integers to bytes and vice versa is reversible. Therefore, it is required to prove that serialization is a bijection.

Lemma 1. *For any integer $x \in \mathbb{Z}$, there exists a bijection between x and a vector $\overrightarrow{z} \in (\mathbb{Z}/256\mathbb{Z})^n$ where $n \geq 1$ such that the components of $\overrightarrow{z}$ are the bytes of x in two's complement big-endian form.*

Proof. It is required to prove that function $f : \mathbb{Z} \to \bigcup_{k\geq 1}(\mathbb{Z}/256\mathbb{Z})^k$ such that $\forall x \in \mathbb{Z}$, $f(x)$ produces a vector $\overrightarrow{z} \in (\mathbb{Z}/256\mathbb{Z})^n$ such that the components represent the bytes that represent x in two's compliment big-endian form. First, recall the process for converting an integer to two's complement form:

- One can find the number of bytes k an integer takes up by evaluating the following formula:

$$k = \left\lceil \frac{\log_2(|x| + 1)}{8} \right\rceil \tag{4}$$

- Then, the value $|x|$ is found and assigned each byte the appropriate value in big-endian form.
- If $x \geq 0$, set the leading bit to 0; if $x < 0$, set the leading bit equal to 1.
- Finally, if $x < 0$, flip each bit of every bit of $|x|$ and add 1 to the resulting value.

Next, it is possible to map every byte of an $8k$ bit integer to a value in $\mathbb{Z}/256\mathbb{Z}$ by corresponding each natural number value to its equivalence class in the byte ring. This yields a vector $\overrightarrow{z} \in (\mathbb{Z}/256\mathbb{Z})^k$.
(Proof of Injectivity): The function is injective if $\exists x_1, x_2 \in \mathbb{Z}$ such that $x_1 \neq x_2$ but $f(x_1) = f(x_2)$. However, by definition, integers have unique representations in two's complement big-endian form. That is, a unique sequence of digits represents any integer z in $\mathbb{Z}_{256}$. If two integers have the same representation in $\mathbb{Z}_{256}$, they must be equal. Therefore, $x_1 = x_2$, and a contradiction is reached. Therefore, f must be an injective function.
(Proof of Surjectivity): Then the proposed function is surjective if every n-byte big-endian vector $\overrightarrow{z} \in (\mathbb{Z}/256\mathbb{Z})^k$ has a representation in the form of an integer. Given vector $\overrightarrow{z}$, an integer can be constructed from the components $z_i \in \mathbb{Z}$ as follows:

- If the most significant bit of $\overrightarrow{z}_0 = 0$: the value of the binary integer is interpreted, and it is possible to calculate the corresponding integer using the formula:

$$x = \sum_{i=1}^{k} z_i * 256^{k-i} \tag{5}$$

- And if the most significant bit of $\overrightarrow{z}_0 = 1$, the corresponding x can be recovered by calculating:

$$x = \sum_{i=1}^{k} z_i * 256^{k-i} - 2^{8k} \tag{6}$$

This reconstructed integer naturally satisfies $f(x) = \overrightarrow{z}$. Therefore, this function must be surjective.
Finally, since f is both injective and surjective, it is, therefore, a bijection. □

Lemma 1 is crucial in establishing the serialization function since it demonstrates that there exists necessarily a bijection between any integer and its byte representation in the byte ring. Therefore, a pairwise execution of the above process for all $x \in \overrightarrow{v}$ should also be bijective. Specifically, since the audio frame is a vector of integers, one can describe the serialization function as a pairwise decomposition of integers greater than 255 into their corresponding byte vectors in the byte ring:

Definition 4. *Define the maximum absolute value of a vector $\overrightarrow{v} \in \mathbb{Z}^n$ as the value:* $\max(|v_i|, \forall v_i \in \overrightarrow{v})$ *and denote this operation by amax($\overrightarrow{z}$).*

Definition 5. *Given a vector $\overrightarrow{v} \in \mathbb{Z}^n$, define the* ***serialization function*** *$S : \mathbb{Z}^n \to (\mathbb{Z}/256\mathbb{Z})^{nk}$ as the pairwise decomposition of each component $v_i \in \overrightarrow{v}$ into its corresponding two's compliment byte vector $b_i \in (\mathbb{Z}/256\mathbb{Z})^k$ where $k = \left\lceil \frac{\log_2(amax(\overrightarrow{v})+1)}{8} \right\rceil$*

By converting each component of an integer vector into its equivalent byte vector, each integer is then efficiently represented in its byte form using the minimal amount of space required. This function, due to its ubiquity in computing, is incredibly efficient to implement and represents the last step in the processing of acoustic samples in embedded recording devices. Serialization represents the last step in the computational work conducted by the microcontrollers before transmission, the exact process of which is entirely contingent on the chosen hardware platform. The simple and linear complexity processes of packaging samples for transmission highlight the degree of minimization of the computational work executed on the microcontrollers. Although the transmission of samples from the microcontroller is efficient, the minimal computations executed necessitate that the processing computer must handle the brunt of the computations. Therefore, the processing computer's algorithmic implementations must be as efficient as possible.

3. Complexity of Storing and Accessing Classification Samples

In this model of an embedded and distributed sensor system, once serialized audio frames have been sent over TCP/IP to the processing computer, it becomes paramount to ensure the efficient recovery, classification, and storage of acoustic samples. The processing computer is responsible for a range of critical tasks: (1) decoding the incoming serialized data, (2) accurately reconstructing the original acoustic samples, (3) classifying them based on predefined criteria, (4) and finally storing them in a way such that relevant samples can be retrieved efficiently for use in other algorithmic processes. First, to ensure the data received by the processing computer is identical to the data transmitted by the microcontrollers, the argument that the serialization S is a bijection is formalized:

Proposition 3. *The **serialization function** is bijective.*

Proof. By definition of the serialization function, each integer is decomposed into its representative byte vector in the byte ring. By Proposition 1, there exists a bijection for each individual conversion $\forall x \in \mathbb{Z}$ to its representation in the byte ring. Now, consider the function serialization function $S : \mathbb{Z}^n \to \mathbb{Z}/256\mathbb{Z}^n * k$ and seek to prove that it is injective in surjective.

- (Injectivity): For injectivity, let us assume for the sake of contradiction that there exist vectors $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{Z}^n$ such that $S(\overrightarrow{x}) = S(\overrightarrow{y})$. Therefore, the byte representation of each element $i \in \overrightarrow{x}$ and each element $j \in \overrightarrow{y}$ must be equivalent by the definition of the serialization function. Recall that the byte representation of two's complement numbers is unique; therefore, $x = y$, a contradiction. Thus, S must be injective.
- (Surjectivity): Assume $\exists \overrightarrow{s} \in (\mathbb{Z}/256\mathbb{Z})^{nk}$. Since there is a bijection for each element $i \in \overrightarrow{s}$ from $\mathbb{Z} \to (\mathbb{Z}/256\mathbb{Z})^k$, there must exist a unique integer $a_i \in \mathbb{Z}$ such that a_i is the integer representation of the i-th byte in $\overrightarrow{s}$. Thus, one can form the vector $(a_1, a_2, ..., a_{nk})$ which by definition equals $S(\overrightarrow{s})$. Therefore, S must be surjective.

Since S is both injective and surjective, S is a bijection. □

This then leads to a natural definition of the deserialization process as the mathematical inverse of the serialization function:

Definition 6. *The **deserialization function** is defined to be S^{-1}.*

Proposition 4. *Given an input vector of length n, the deserialization function is $O(n)$*

Proof. Recall the process of converting an integer into its two's complement form as described in Lemma 1. All the operations in the process of converting an integer to its two's complement form are $O(1)$ and are implemented efficiently on all modern hardware architectures. Since the process is composed of a constant number of operations regardless, converting a single integer to its two's complement form must be $O(1)$. Since for a vector of length n, this conversion needs to be repeated n times; therefore, the deserialization function must be $O(n)$. □

With the ability to efficiently reconstruct the audio frame data structure on the processing computer, the next step proposed in processing the acoustic samples is to create a dataset on which dense neural networks are trained. Recall that in transfer learning, each waveform must be passed through YAMNET to obtain an embedding. The conversion of acoustic samples to numeric vectors by YAMNET is considered a black box function denoted by $Y : \mathbb{Z}^n \to \mathbb{R}^{2048}$ where n is the length of the input vector to the black-box function. Therefore, when L is the set of labels on which the neural network can classify, and $\exists y_1, y_2, ..., y_n \in L$, a dataset of n acoustic samples $\overrightarrow{a}_1, \overrightarrow{a}_2, ..., \overrightarrow{a}_n$ can be formulated as follows:

$$D = \{(Y(\overrightarrow{a}_1), y_1), (Y(\overrightarrow{a}_2), y_2), ..., (Y(\overrightarrow{a_n}), y_n)\} \tag{7}$$

This dataset pairs each high-dimensional embedding with its corresponding label, effectively capturing the features needed for classification with a DNN. Once the dense neural network classifies the acoustic samples, storing positive classification results efficiently becomes essential. Practically, several methods are viable for storing positively classified samples, from databases to priority queues. For the sake of simplicity, assume that positive results are stored in a list for use. Then, an efficient search algorithm exists to access the most recent acoustic samples:

Proposition 5. *Let $x = \{1, ..., ...n\}$ be the list of IDs for each sensor in the detection system, and a list of id, timestamp tuples $l = ((x_1, y_1), (x_2, y_2), ..., (x_m, y_m))$. It takes $O(m \log m)$ time using a two-pointer-based approach to find a subset of n tuples, one for each x value, such that the difference between the maximum and minimum timestamp (y) values among the selected tuples is minimized.*

Proof. First, sort the list l (in ascending order) with respect to the timestamps $y_1, ..., y_m$. The complexity of this operation is on the order of $m \log m$ where m is the length of the list. Without loss of generality, assume the sorted list is indexed identically to the pre-sorted list, that is sorted$(l) = l'((x_1, y_1), (x_2, y_2), ..., (x_m, y_m)), y_1 \leq y_2 \leq ... \leq y_m$. Now, assume there exist variables left and right, whose values are initially 1, and a variable diff representing the maximum time difference between timestamps in the sublist spanned between indexes left and right. Where l'_i represents the i-th element of l', the algorithm is as follows:

Require: List of elements $l'_1, \ldots, l'_m$ with labels, and all possible labels x
Ensure: Minimum difference `diff`

Initialize `diff` to a large value, usually maximum integer value
Set `left` = 1, `right` = 1
while `right` $\leq m$ **do**
    while the sublist $(l'_1, \ldots, l'_{\text{right}})$ does not contain all possible labels x **do**
        Increment `right` by 1
    end while state Compute the current difference:

$$\texttt{cur\_diff} = l'_{\text{right}}.y - l'_{\text{left}}.y$$

    if `cur_diff` < `diff` **then**
        Update `diff` = `cur_diff`
    end if
    while incrementing `left` maintains a sublist containing all labels x **do**
        Increment `left`
    end while
end while

In the worst case, there are $2m$ reassignments of the pointer variables left and right, indicating that this portion of the algorithm is $O(m)$. The algorithm takes $O(m \log m + m)$ time, counting the initial sort, indicating that its time complexity is $O(m \log m)$. □

The proposed algorithm is particularly relevant in multilateration, where each element in the sublist corresponds to a timestamp from each sensor. Selecting positive classification samples with as little time difference as possible permits increased fidelity in the estimates by a theoretical multilateration system. Moreover, the ability to flag the recent classification samples likely has applications in other processes, from determining whether a particular sample is a false positive to cross-validation with other detection systems to ensure the accuracy of results. The finalization of the algorithm marks a significant advancement toward integrating this process into broader security systems or computational frameworks. The algorithms proposed in the past two sections demonstrate the process of efficiently gathering and processing acoustic samples, from their initial acquisition at the sensor level to their sorting and labeling for more extensive algorithmic processes. These developments demonstrate the viability of the proposed approach and provide a foundation for physical implementations of the detection system.

C. Prototype Models for UA Detection

1. Arduino-Based UA Detection Prototype

The Arduino-based detection system proposed in this section was an initial attempt to create a device solely to record UAs in flight. The samples the Arduino gathered were then used to generate a dataset to train neural networks to detect the presence of UAs. Furthermore, using an Arduino Mega 2560 represented the lower bound for possible hardware constraints in a detection environment. Specifically, the microcontroller is only capable of transmitting a single 8000 sample audio frame per second, which is then upsampled to 16kHz and converted into a floating-point encoding format to fit within the requirements posited by YAMNET [16]. The high-level process of sample acquisition is visualized in Figure 3. This initial prototype led to the first implementations of the audio frame data structure as well as algorithms for its serialization. Overall, the testing of this system enables the determination of whether the algorithmic processes described in Section II.B.2 would hold up to the requirements of real-time classification as well as create the transfer learning dataset for UA detection. The per-unit cost of the sensor is under $100, consisting of an Elegoo Mega 2560 R3 Board, the Arduino Ethernet Shield 2, and a single MAX4466 analog microphone. These parts are all consumer-grade and are acquirable from consumer retail websites, such as Amazon. This sensor system then connects to the laptop and processes the samples it receives from the Arduino via a TCP/IP connection over Ethernet.

Acoustic data was successfully detected in the field on four different days using this detection system, and the samples gathered during field testing comprise the UA Detection Dataset [17]. Its success in uploading samples to the laptop and collecting over 8,000 samples demonstrated the viability of this prototype for capturing acoustic samples for classification. Despite this, testing revealed some problems with the construction of this system:

- High Unit Cost: The use of an Arduino contributes to significant expense despite its minimal computational power compared to other microcontrollers. For instance, the ATMega chip present on the Mega 2560 is only an 8-bit processor, while the majority of other microcontrollers are 32-bit. The Arduino Mega 2560 also only has 8KB of ram, significantly less than competing microcontrollers such as the ESP32, which has 520KB of RAM and can be expanded using its built-in SPI bus.
- Wired Connection Dependency: Due to the lack of commercially available hardware for the Mega 2560, it must always be connected to the laptop for both power and transmissions.
- Limited Sampling Rate: The built-in analog-to-digital converters on the Arduino Mega 2560 can not sample above 8KHz, making it impractical to gather high-fidelity recordings using them directly.

For these reasons, the development of a second UA detection system was needed, one that could operate wirelessly and be portable to permit the testing of multilateration.

2. ESP32-Based UA Detection Prototype

To remedy the faults of the previous detection system, the authors constructed a multiple-sensor iteration of the proposed UA detection model, using ESP32s as a microcontroller base and digital MEMS microphones instead of analog microphones. This prototype corresponded to the first attempt at conducting classification from multiple devices simultaneously and transmitting data over Wi-Fi as opposed to the Ethernet used in the original prototype. The sample acquisition and processing are then described in Figure 4. The per-unit sensor cost is also significantly less than the first system, totaling under $20 per unit for the ESP32, IMNP441 microphone, and portable battery used to power the sensors. However, the additional one-time unit cost of acquiring a router, which was purchased for $20, increasing the amortized per-unit sensor cost to around $25. This is over a 50% drop in per-sensor cost compared to the Arduino-based prototype while providing three key benefits over the original system: (1) the ability to place sensors in arbitrary locations on the field due to being battery-powered, (2) having significantly more computational power, (3) and having built-in Wi-Fi networking on-chip. These benefits certainly make the ESP32 a far more viable platform for acoustic sample acquisition.

In the realm of sample acquisition, to further validate the effectiveness of this ESP32-based UA detection system, multiple field tests are conducted where the prototype's functionality and resilience are evaluated under realistic conditions. During these tests, each sensor unit is configured to operate independently and transmit data to the laptop for real-time processing while communicating wirelessly and operating solely off of batteries. Testing verified that all

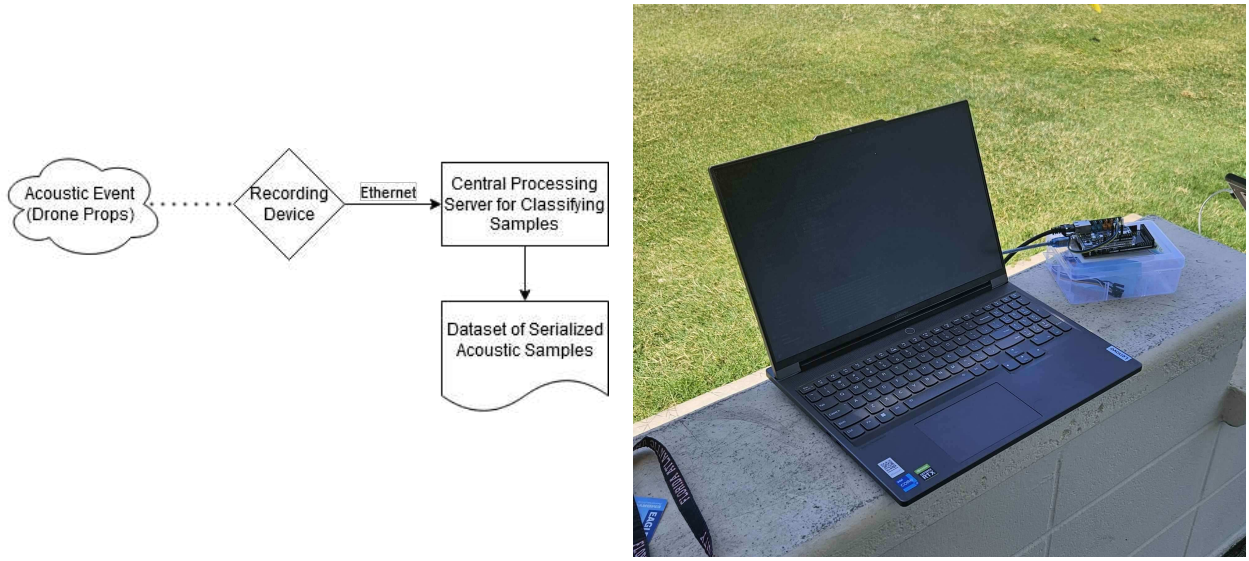

Fig. 3 Workflow diagram of the Ardino detection model (Left) and photo from field testing (Right).

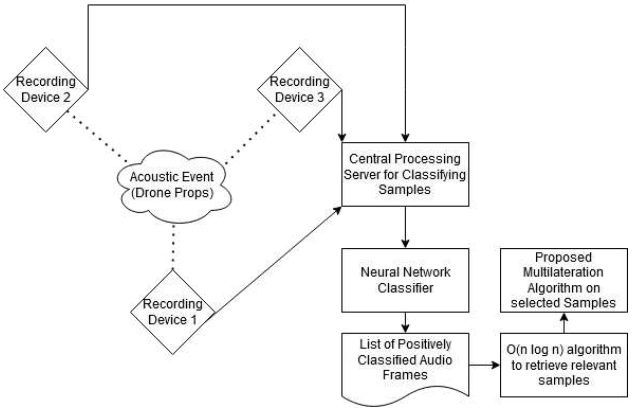

Fig. 4 Workflow used in testing multiple recording devices and multilateration

three sensors reliably capture and transmit audio samples over Wi-Fi simultaneously without packet loss, maintaining data integrity and synchronization across devices and demonstrating the ability of the UA detection system to capture inputs from multiple devices simultaneously. Although an extensive statistical analysis of this system is out of the scope of this research, preliminary results of this prototype's applications to multilateration are provided and discussed below.

D. Training of Neural Networks for UA Classification

As described above, all received acoustic samples are processed through Google's YAMNET to conduct transfer learning. The embeddings for each single-second acoustic sample are then extracted and saved as a serialized tensor file for later retrieval. The embeddings were also archived and uploaded to the Embry-Riddle Aeronautical University Data Commons for long-term storage and accessibility [17]. This prototype classification system is tested on a soccer field over four days, where the UA ranged from 10 feet to more than 100 feet away. This soccer field is also near Daytona Beach International Airport, making the background noisier and creating more variability in the dataset. Varied wind conditions on different days in the field also contributed to more diversity in the received samples *. The dataset contains samples from multiple types of UAs, permitting UA-type classification. UAs recorded include the DJI Mavic 3, DJI Mavic Mini, and DJI Matrice M100. The types of UAs are differentiated since they all have different prop radii and diameters, making for different audio profiles. Negative data samples were collected by capturing various sounds, from cars to airplanes, to provide a broad basis of samples for the neural network to classify over. In this stage, the detection system gathered 8,864 embeddings, which comprise the UA dataset used to evaluate the neural network.

In addition to detecting the three UA types mentioned previously, samples of the Matrice M100 with and without payloads attached to it have been acquired, as seen in Figure 5. This was done to increase the models' versatility in detecting UAs since the flight profile of the M100 with a payload is different from one without. The acquisition of varied acoustic samples permits further generalization of neural networks to a wider range of UA acoustic signatures, helping to prevent overfitting.

Two different neural networks have been trained based on the dataset of embeddings gathered: (1) a binary UA/No UA classification model and (2) a model meant to classify different types of UAs. The first network uses the YAMNET embeddings to perform a binary classification between the presence of a UA in the embedding and the lack of one. In a practical security system, this network conducts the initial classification, and if it receives a positive sample, it triggers other algorithms to gather more information about the intrusion. The second neural network then classifies the type of drone based on the sub-labeled embeddings. The two neural networks are nearly identically structured, taking embedding vectors from YAMNET as input. As seen in Figure 6, the samples are then passed through a 50% dropout layer, a batch normalization layer, another 50% dropout layer, and finally into a softmax layer representing the final layer of classification. The binary model has two neurons on this layer, and the UA-type classifier has three, representing the total number of classes each model classifies. The networks used the Adam Optimizer with a learning rate of 0.05 and the categorical cross-entropy loss function. The data was split 80% for training, 10% for testing, and 10% for validation. To assess the performance of the trained neural networks, the following metrics are utilized:

*High wind can cause interference on microphones, and windscreens may be a worthwhile investment on a production system.

Fig. 5 Photograph of the DJI Matrice M100 (Left) and DJI Mavic 3 (Right).

- Validation Accuracy: The proportion of correctly classified samples in the validation dataset, indicating the model's ability to generalize.
- Precision: The ratio of correctly classified samples to the total number of samples classified as true. This metric gives the frequency that a positive sample is classified as such.
- Recall: The ratio of correctly classified samples and the total number of samples labeled as positive. This metric returns the accuracy of a model returning
- F1 Score: The harmonic mean of the model's precision and recall, providing a balanced measure of the model's performance.
- Loss Function Derivatives: Analyzing the convergence and rate of change of the loss function can provide insights into whether a neural network model is overfitting or not.
- Confusion Matrices: A confusion matrix is a table that displays the count of true positives, true negatives, false positives (Type I Errors), and false negatives (Type II Errors). These metrics provide insight into the prediction accuracy and type of errors classification models make.

These metrics allow for a detailed analysis of both generalization capability and class-specific performance. Finally, these metrics are also analyzed over multiple iterations of the neural networks' training, mitigating the differences caused by randomness in the training process. These foundations serve as the methods of statistical analysis of the networks discussed in the following section and allow for a thorough determination of the practicality of the trained neural networks.

III. Results

In this section, we present the results of our model training experiments, reviewing the performance of both the proposed binary and UA-type classification models. Both models were trained five separate times, each time being trained over ten epochs, also known as passes over the entire training dataset. After training, the binary classification model had a validation accuracy averaging 95.372% with a standard deviation of 0.965%. In contrast, the UA-type classification model had validation accuracies of 99.185% and a standard deviation of 0.569%. Due to the similar training and validation accuracies present in both models, as well as the monotonic decrease in training loss (Figure 7), neither model likely overfitted during the training process. Moreover, the binary classification model had an average F1 score of 0.954 over its five different training iterations, representing that overall, the model is effective at classifying UAs from its surrounding environment. The recall for the positive and negative classification classes is also relatively high at 0.904 and 0.982, respectively. On the other hand, the lower recall of the positive classification class paired with a relatively high standard deviation of 0.0439 likely indicates that some samples in the positive classification class might

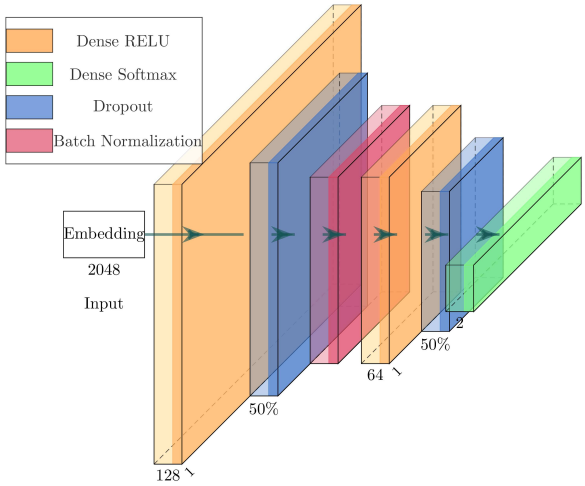

Fig. 6 Neural network diagram for the binary classification neural network

be near the decision boundary for the model. With respect to the authors' previous work, the model's average accuracy exceeds the average accuracy of 86% when identifying the presence of a UA [8]. Considering the binary confusion matrix in Figure 8, the increased instances of false negatives further elucidate that there may be specific samples in the positive sample class that the binary-classification model is struggling to classify, such as samples gathered when the UA is taking off or landing; or when the UA is at the edge of the acoustic sensor's recording range. This result may also be attributable to the fact that the underlying dataset is somewhat unbalanced, having 3198 positive acoustic samples and 5666 negative acoustic samples.

In contrast, the UA-type classification model maintained a very high average F1 score of 0.99. The Matrice M100, Mavic 3, and Mini 2 classes all had similar mean recalls of 0.998, 0.978, and 0.982, and similarly low standard deviations of 0.0044, 0.0148, and 0.0179, respectively. These metrics indicate that the UA-type model likely did not struggle with any particular class. However, it clearly performed best when identifying the Matrice M100 compared to other classes. This also may be due to an imbalance in the dataset, for there are a total of $1,900$ samples for the Matrice M100, but only 655 for the Mavic 3 and 643 for the Mini 2. The higher frequency of misclassifications of the Mavic 3 and Matrice M100 compared to other errors in Figure 8 also demonstrates the potential imbalance between the sizes of the Matrice M100 and Mavic 3 classes. In future iterations of this model, it would be worthwhile to gather more samples from the Mavic 3 and Mini 2 drones.

Together, these results demonstrate that the classification model is highly effective in detecting and identifying UAs using acoustic data. The high validation accuracies and low standard deviations across multiple training iterations confirm the model's reliability and robustness, while the tendency for errors to be Type I instead of Type II in the binary classification model demonstrates that the networks are unlikely not to detect a drone if it enters the range of acoustic sensors. Moreover, the successful deployment of the model on a microcontroller with limited memory underscores its practicality for embedded systems. This work not only validates the theoretical foundations laid out in this paper but also serves as a practical basis for constructing efficient UA detection systems capable of operating in resource-constrained environments.

IV. Discussion

A. Theoretical Nuance between SVMs and Neural Networks

In previous work, acoustic signatures of drones were gathered, as well as spectral descriptors from short (5-second) stints of acoustic data. Spectral descriptors are a class of statistical notions primarily based on central moments and entropy that describe the frequency content and distribution of acoustic signals. For instance, centroids are analogous to the first central moment of an acoustic signal and represent the gravitational center of a spectrum, exposing the dominant frequencies present in a waveform, while metrics such as spectral entropy evaluate how the frequency content of a sound changes over time [8]. For each sampled time interval, one would then "package" these sounds in a vector

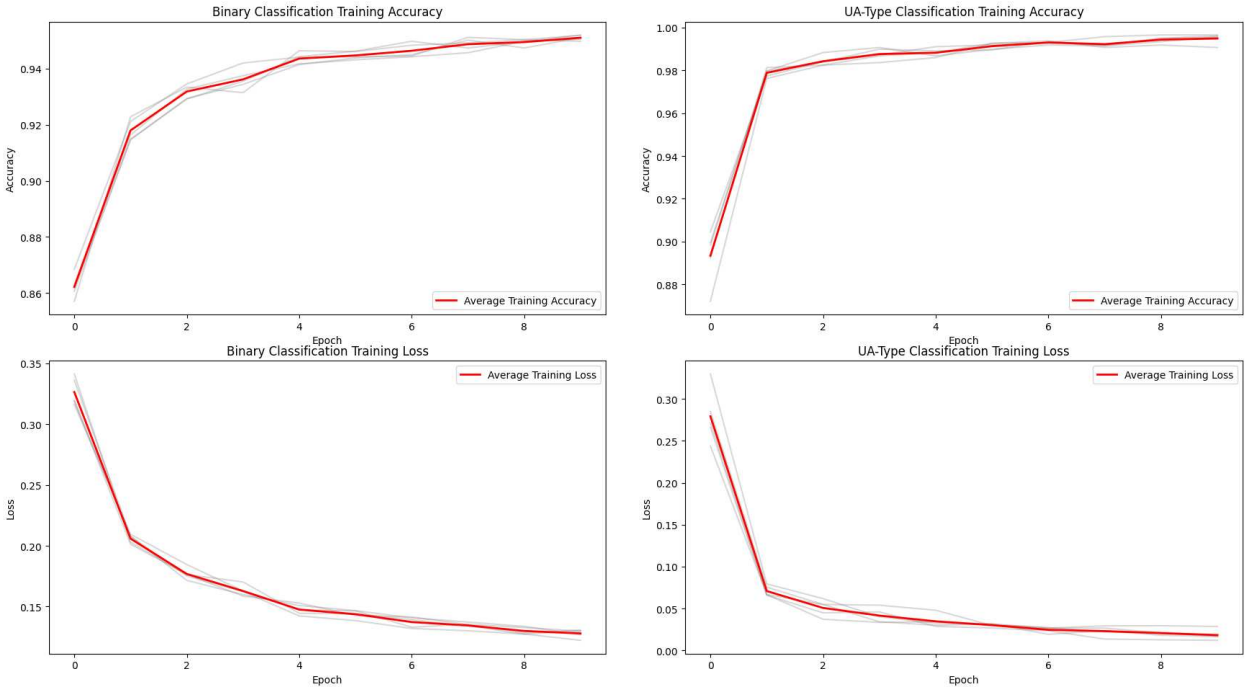

Fig. 7 Training accuracy and losses averages (Red) and individual iterations (Gray) over five training runs for UA classification dataset with sample sizes of 8864 **and** 3198 **respectively.**

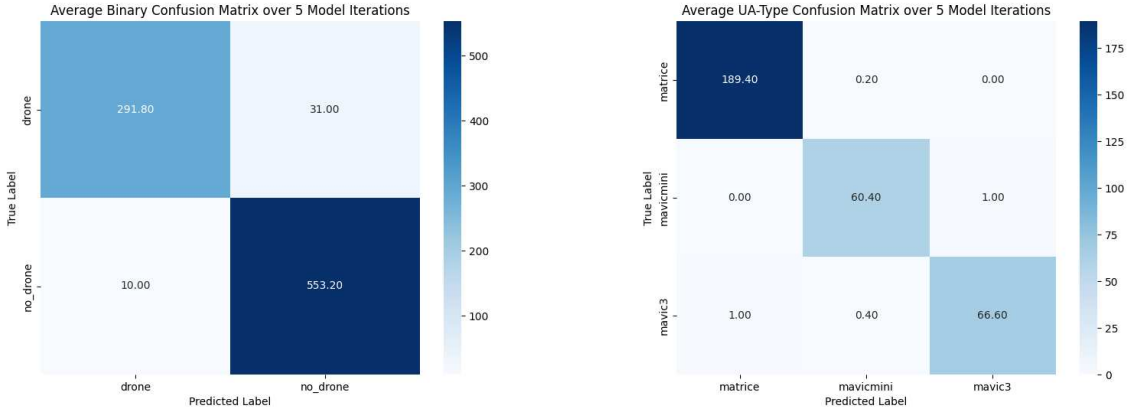

Fig. 8 Confusion Matrices over Validation Datasets for both Binary and UA-Type Classifiers Respectivly

$\overrightarrow{X} = [x_1, x_2, ..., x_n]$ where each component $\overrightarrow{x_i}$ represents a spectral descriptor. To then classify for the presence of UAs, each sample would be assigned a label $y_i \in \{-1, 1\}$ corresponding to a negative and positive result. The overarching goal of binary classification SVMs is to find a hyperplane that divides the two classes of data with the largest margins possible. This procedure stands in contrast with neural networks, which are composed of a series of neurons that output a weighted combination of inputs they take. These simple models can then be grouped into layers and learn by tuning the weights of each neuron to emulate some unknown function that represents the dataset D. In a mathematical sense, a neural network attempts to approximate some unknown function $f*$ by adjusting the parameters to the function $f_\theta(\overrightarrow{x}^{(1)})$ that represents the output of the entire neural network.

These two approaches represent attempts to manually and automatically extract features from the data, respectively. In constructing a distributed UA detection system, the computational cost of extracting statistical features from the acoustic waveforms in the SVM model must be left to the microcontrollers, which is difficult due to memory and

computational constraints, or the acoustic waveforms must be uploaded to the processing computer, which provides no bandwidth benefits over the CNN-based detection model. Therefore, the use of CNNs is proposed here to automatically learn and extract relevant features from acoustic waveforms received from the distributed sensors.

B. Disparity Between Binary and Type-Classification Model Accuracy

Two results that immediately stand out are that the UA-type classification model performed 3.81% better than the binary classification model. Naturally, it seems intuitive that the minute differences between the sounds of multiple UAs would be significantly more challenging than differentiating an unrelated sound and a UA flying. A potential two-fold cause is suspected for the disparity in the performance of these models: (1) due to the assumption of positive UA presence in the samples of the UA-type classifier and (2) the different conditions in which some UA types were captured. One significant advantage the UA-type classifier has over the binary classification network is the assumption that all the samples it receives have UAs present in the waveforms. This may provide an advantage since the binary classifier considers a much more comprehensive range of samples, from jets taking off to the hum of cars to actual UAs. This indicates that the final dense network may need to consider a much larger range of values in the embedding arrays, and the weights of the network may be similarly varied. Additionally, samples for Mavic 3 and Mini 2 were only able to be captured in the laboratory due to airspace restrictions present due to the proximity of the Embry-Riddle Aeronautical University to Daytona Beach International Airport. Different background noise in the laboratory setting as opposed to in the field may introduce some bias into the classification. Despite this, the UA-type network has similar fidelity in differentiating the Mavic 3 and Mini 2 samples as it does with them and the Matrice, demonstrating that this is not the primary reason for the UA-type classifier's outperformance of the binary classifier.

C. Differences between Analog and MEMS microphones for Acoustic Sensors

In the development of the neural networks presented in the papers, multiple iterations of recording devices for the purpose of collecting data and testing multilateration have been developed. The primary difficulty faced with the use of analog microphones is finding microcontrollers that have analog-to-digital converters (ADCs) that could sample the analog microphone at a rate appropriate for classification. According to Nyquist's Theorem, the frequency range efficiently sampled by a microphone is less than half of its sample rate. For example, a microphone sampled at 8KHz can capture up to 4KHz worth of frequency range without aliasing, a type of interference. For the purpose of UA classification, we consider an 8KHz sample rate to represent a floor for the fidelity required for classification.

Among mass-produced consumer-grade microcontrollers, used because of their lower cost compared to specialty boards and hardware, few have ADCs that sample at rates over 8KHz. For example, the Arduino Mega 2560 can only sample at $\approx$ 9.6 KHz at its default clock speed, and the ESP32 can only sample its built-in ADC at 6 KHz. Digital microphones, such as MEMS microphones, utilize the I^2S interface, which permits the sampling of left/right channels as well as sampling each at an arbitrarily high frequency. The downside of consumer-grade MEMS microphones in testing is their greater sensitivity to noise and reduced range compared to analog microphones. Sampling analog microphones at arbitrarily high rates, like most MEMS microphones, can be accomplished using dedicated integrated circuits that convert an analog input to an I^2S stream. This would likely achieve benefits brought forth by both approaches with the downside of the additional cost of a specialized integrated circuit.

D. Viability of Multilateration in Proposed Detection model

One of the primary motivations for constructing a distributed detection system for UA detection is to attempt to use time-difference of arrival (TDOA) multilateration to locate a drone in uniform time steps. In testing, the processing computer is found capable of processing and conducting both binary and type classification on acoustic samples in real-time. Then, to assess the most effective approach among the multiple optimization algorithms presented in most machine learning libraries, a theoretical model of multilateration was developed in Python. This model simulates the process of using multiple distributed sensors to determine the precise location of an acoustic event by analyzing the time differences of arrival (TDOA) of the sound at each sensor. Then, the Nelder-Mead method was tested along with the Powell Algorithm, the Broyden-Fletcher-Goldfarb-Shanno algorithm, the Constrained Optimization by Linear Approximations (COBYLA), and the Sequential Least-Squares Programming optimizer. Their performance is then evaluated by placing an acoustic event at an arbitrary point and evaluating which models get the closest approximation to the actual position of the event (Figure 9).

Generally, it has been found that COBYLA consistently outperformed all the other aforementioned optimizers in its

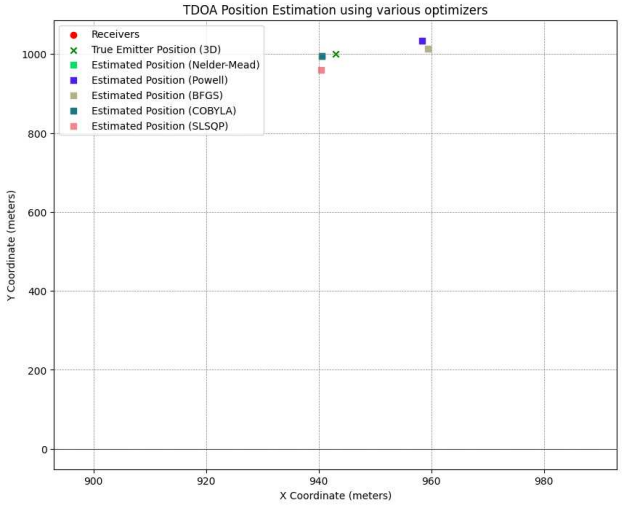

Fig. 9 Performance of optimizers when sensors are in an equilateral triangle formation of length 1000M with a sensor at (0, 0, 0) and event position of (943, 1000, 50).

accuracy in pinpointing an acoustic source. Thus, we consider it the best candidate for use in a TDOA multilateration algorithm. COBYLA's performance is likely due to the task of multilateration boiling down to finding the approximate solutions to a system of hyperbolas formed between the time of arrival differences between each center, thus making non-linear optimizers more viable than linear ones. However, COBYLA is an unimodal optimizer, and for more complicated systems where there may be multiple local minima, a multimodal optimizer may prove to be a better choice. Another factor to consider in the implementation of multilateration for a UA detection system is the distance between sensors. When sensors are positioned too close to each other, errors due to unavoidable latency in sample acquisition and transmission become more pronounced. For example, with a timestamp error of up to 5 milliseconds, readings from sensors spaced within 11.25 feet of each other could be rendered unreliable. Therefore, efforts to reduce error as much as possible in acquired timestamps could be incredibly valuable in increasing the accuracy of multilateration. Overall, the ability of the processing computer to process inputs from multiple sensors paired with the insights gained from the theoretical model demonstrates that our proposed UA detection model can likely be applied to multilateration.

E. Future Work

The success in the accuracies of the neural networks demonstrates the potential for further development of the proposed acoustic UA detection system. There are many places this system can be refined, whether in the acoustic sensors themselves, as discussed above, in the development of fully-functioning multilateration, or in validating the security of the system against attackers. Particularly in the realm of multilateration, much algorithmic development can occur: (1) ensuring the synchronization of timestamps between the ESP32s to reduce error in multilateration and (2) development of cross-correlation algorithms to detect shared peaks between the audio frames of each sensor. These developments should serve as a sufficient basis for a fully-functioning UA detection system based on the ESP32 prototype. On the other hand, a critical consideration in deploying the ESP32-based detection system is the security of data transmission over Wi-Fi. Unlike the wired Ethernet connections used in the previous prototype, wireless communications make wiretapping and jamming significantly simpler. To mitigate these risks, secure encryption protocols, such as WPA3, should be used to protect the integrity of the data transmitted and ensure non-repudiation of the acoustic samples. Furthermore, attacks by actors such as Wi-Fi jamming may have the potential to compromise this system. For these reasons, consideration of anti-jamming techniques or alternate wireless communication methods may aid in increasing the resilience of this system against malicious actors.

V. Conclusions

Overall, two significant algorithmic developments were realized to make a neural-network-based mode of UA detection possible: (1) the development of an efficient algorithmic pipeline to offload acoustic data from embedded

devices as soon as it is gathered and (2) the use of transfer learning to utilize neural networks for feature extraction. Furthermore, the centralization of acoustic sample processing and the use of transfer learning make this mode of embedded and distributed classification viable on consumer-grade hardware, which significantly reduces the cost of UA detection systems. These improvements in classification accuracy, combined with a drop in the cost of acoustic sensors associated with the use of consumer-grade hardware, demonstrate the potential of these systems for broader applications and justify their further development and refinement.

Then, to evaluate the performance of proposed binary and UA-type classification models, we conducted an experiment in which both models were trained five times and their performance metrics analyzed. Overall, this analysis revealed that both models demonstrated high levels of recall and found a jump in classification accuracy of 9% between previous and current work. These metrics represent the potential of the feature extraction of CNNs in the realm of UA detection. These results are likely due to the ability of CNNs to train their own feature detection algorithms, while previous work relies on spectral descriptors and statistical notions to capture the essence of the waveform. Ultimately, these developments mark a significant step towards reliable and cost-effective UA detection systems, with the potential to significantly improve aerial surveillance capabilities.

VI. Acknowledgements

This effort is based on work supported in part by the National Science Foundation under Grant number 2050887. Assistance from colleagues in the Space Trajectories and Applications Research group at Embry-Riddle Aeronautical University is acknowledged, as is the support from the Aerospace Engineering department. The opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Shepardson, D., "Drone sighting briefly disrupts flights at Pittsburgh airport," *Reuters*, 2023. URL https://www.reuters.com/world/us/drone-sighting-disrupting-flights-pittsburgh-airport-2023-06-05/.

[2] Koury, R., Jul. 2023. URL https://abc7news.com/homeowners-insurance-non-renewal-drone-photos-nonrenewal/13519714/.

[3] Claburn, T., "Wi-Fi spy drones used to snoop on financial firm," , 2022. URL https://www.theregister.com/2022/10/12/drone-roof-attack/.

[4] Administration, F. A., "14 CFR Part 89: Remote Identification of Unmanned Aircraft," , 2021. URL https://www.ecfr.gov/current/title-14/part-89.

[5] Seidaliyeva, U., Ilipbayeva, L., Taissariyeva, K., Smailov, N., and Matson, E. T., "Advances and Challenges in Drone Detection and Classification Techniques: A State-of-the-Art Review," *Sensors*, Vol. 24, No. 1, 2024. https://doi.org/10.3390/s24010125, URL https://www.mdpi.com/1424-8220/24/1/125.

[6] Wallace, R. J., Rice, S., Lee, S.-A., and Winter, S. R., "Unveiling Potential Industry Analytics Provided by Unmanned Aircraft System Remote Identification: A Case Study Using Aeroscope," *Drones*, Vol. 8, No. 8, 2024. https://doi.org/10.3390/drones8080402, URL https://www.mdpi.com/2504-446X/8/8/402.

[7] Tedeschi, P., Nuaimi, F. A. A., Awad, A. I., and Natalizio, E., "Privacy-Aware Remote Identification for Unmanned Aerial Vehicles: Current Solutions, Potential Threats, and Future Directions," *IEEE Transactions on Industrial Informatics*, Vol. 20, 2024, pp. 1069–1080. URL https://api.semanticscholar.org/CorpusID:259819449.

[8] Patel, K., Ramirez, L., Canales, D., and Rojas, E., "Unmanned Aerial Vehicles Detection Using Acoustics and Quantum Signal Processing," *AIAA SCITECH 2024 Forum*, American Institute of Aeronautics and Astronautics, Orlando, FL, 2024. https://doi.org/10.2514/6.2024-1740, URL https://arc.aiaa.org/doi/10.2514/6.2024-1740.

[9] Shi, Z., Chang, X., Yang, C., Wu, Z., and Wu, J., "An Acoustic-Based Surveillance System for Amateur Drones Detection and Localization," *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 3, 2020, pp. 2731–2739. https://doi.org/10.1109/TVT.2020.2964110.

[10] Bachmann, S., DeBrunner, V., and Zrnic, D., "Detection of Small Aircraft with Doppler Weather Radar," *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, 2007, pp. 443–447. https://doi.org/10.1109/SSP.2007.4301297.

[11] Kelly, E. J., Reed, I. S., and Root, W. L., "Early Advances in Radar Technology for Aircraft Detection," *Lincoln Laboratory Journal*, 2001. URL https://api.semanticscholar.org/CorpusID:16040201.

[12] Lu, Y., Wang, Z., Tang, Z., and Javidi, T., "Target Localization with Drones using Mobile CNNs," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2566–2573. https://doi.org/10.1109/IROS.2018.8594163.

[13] Nalamati, M., Kapoor, A., Saqib, M., Sharma, N., and Blumenstein, M., "Drone Detection in Long-Range Surveillance Videos," *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019, pp. 1–6. https://doi.org/10.1109/AVSS.2019.8909830.

[14] Taha, B., and Shoufan, A., "Machine Learning-Based Drone Detection and Classification: State-of-the-Art in Research," *IEEE Access*, Vol. 7, 2019, pp. 138669–138682. https://doi.org/10.1109/ACCESS.2019.2942944.

[15] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, Vol. 25, edited by F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[16] Google, "YAMNet," https://www.kaggle.com/models/google/yamnet/tensorFlow2/yamnet/1?tfhub-redirect=true, 2020. URL https://tfhub.dev/google/yamnet/1, version 1, TensorFlow Hub model.

[17] Stafford, V., and Canales Garcia, D., "Uncrewed Aircraft Detection from YAMNET Embedding Dataset," , 2024. https://doi.org/10.17632/5DMCSZVYM4.2, URL https://data.mendeley.com/datasets/5dmcszvym4/2.