


Article

Deep-Learning-Based Stream-Sensing Method for Detecting Asynchronous Multiple Signals

Yeongjun Kim ¹ and Harim Lee ^{2,*} ¹ Department of Electrical Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan 44919, Korea; kimyj783@unist.ac.kr² School of Electronic Engineering, Kumoh National Institute of Technology, Gumi-si 39177, Korea

* Correspondence: hrlee@kumoh.ac.kr

Abstract: In a disaster site, terrestrial communication infrastructures are often destroyed or malfunctioning, and hence it is very difficult to detect the existence of survivors in the site. At such sites, UAVs are rapidly emerging as an alternative to mobile base stations to establish temporary infrastructure. In this paper, a novel deep-learning-based multi-source detection scheme is proposed for the scenario in which an UAV wants to estimate the number of survivors sending rescue signals within its coverage in a disaster site. For practicality, survivors are assumed to use off-the-shelf smartphones to send rescue signals, and hence the transmitted signals are orthogonal frequency division multiplexing (OFDM)-modulated. Since the line of sight between the UAV and survivors cannot be generally secured, the sensing performance of existing radar techniques significantly deteriorates. Furthermore, we discover that transmitted signals of survivors are unavoidably asynchronous to each other, and thus existing frequency-domain multi-source classification approaches cannot work. To overcome the limitations of these existing technologies, we propose a lightweight deep-learning-based multi-source detection scheme by carefully designing neural network architecture, input and output signals, and a training method. Extensive numerical simulations show that the proposed scheme outperforms existing methods for various SNRs under the scenario where synchronous and asynchronous transmission is mixed in a received signal. For almost all cases, the precision and recall of the proposed scheme is nearly one, even when users' signal-to-noise ratios (SNRs) are randomly changing within a certain range. The precision and recall are improved up to 100% compared to existing methods, confirming that the proposal overcomes the limitation of the existing works due to the asynchronicity. Moreover, for Intel(R) Core(TM) i7-6900K CPU, the processing time of our proposal for a case is 31.8 milliseconds. As a result, the proposed scheme provides a robust and reliable detection performance with fast processing time. This proposal can also be applied to any field that needs to detect the number of wireless signals in a scenario where synchronization between wireless signals is not guaranteed.

Keywords: UAV-based rescue system; deep-learning-based system; asynchronous stream sensing**Citation:** Kim, Y.; Lee, H.Deep-Learning-Based Stream-Sensing Method for Detecting Asynchronous Multiple Signals. *Appl. Sci.* **2022**, *12*, 4534. <https://doi.org/10.3390/app12094534>

Academic Editor: Douglas O'Shaughnessy

Received: 30 March 2022

Accepted: 27 April 2022

Published: 29 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

According to [1], the statistical graph titled “Number of accident reports by type, 1970 to 2019” shows that natural disasters are on the rise. In the 20th century, large-scale natural disasters steadily occurred, including the Great Earthquake in Haiti in 2010, the tsunami caused by the Great East Japan Earthquake in Japan in 2011, and the Maria Hurricane in the United States in 2017. These large-scale natural disasters inevitably result in a huge number of casualties. To rescue more survivors and minimize the use of human and monetary resources, it is very important to accurately detect the existence of survivors in a certain coverage. Existing wireless infrastructures could be of great help in confirming the existence of survivors. However, conventional wireless infrastructures are likely to be destroyed

or out of order at a disaster site, and thus it is necessary to develop a cost-effective but powerful system that can detect the existence of survivors at such a site.

Due to its flexible operation capability and cost-effectiveness, UAVs can be a perfect candidate for an autonomous survivor search system. Specifically, rescue UAVs can be used for two purposes: (1) recovery, and (2) search and rescue. Recently, for the recovery of the telecommunications infrastructure, several major telecommunications companies such as AT&T and Verizon started to evaluate the possible benefits of using UAV base stations [2,3]. These attempts have confirmed that UAV base stations can recover damaged infrastructure costeffectively. In contrast, despite its high potential, there have been few efforts using UAVs for survivor search and rescue purposes.

A problem that always arises in operating an UAV is the UAV's power consumption. If power is supplied by wire, there is no problem with the operating time, but the operating radius is limited due to the power wire. It is exactly the opposite when UAVs are operated through batteries.

Despite limited operating time, battery use is essential in a disaster site as a rescue UAV has to investigate a large disaster area. To reduce the power consumption of battery-powered UAVs, the payload should also be minimized. Therefore, a computationally lightweight survivor-detection algorithm should be developed to avoid the use of high-end companion computers, which are usually heavy and also consume a lot of power. Furthermore, since the global market size for small UAVs is expected to reach USD 22.55 billion by 2026 and to grow at a CAGR of 15.92% [4], it is more likely that survivor detection UAVs will be small, with minimal payload.

To find survivors, an UAV can utilize a high-resolution camera to perform object-detection algorithms. However, this method has the limitation that if survivors are obscured by obstacles, an UAV cannot find them. Alternatively, it is possible to estimate the number of survivors by obtaining communication history from the telecommunications companies. However, as rescue time increases, this information becomes outdated, since survivors can move around the disaster area, which can confuse rescuers. Therefore, there is a need for a technology that can continuously update the number of survivors in real time without being disturbed by obstacles.

Therefore, it is necessary to develop a lightweight and effective search scheme for survivors. In this paper, we develop a survivor search system that consists of a machine learning-based multi-source detection scheme that detects the number of simultaneously transmitted signals from survivors' mobile devices and a simple survivor discovery protocol. For the survivor discovery protocol, there is a challenge that an UAV should detect the number of signals that are transmitted *asynchronously*. For the solution, a machine learning-based asynchronous stream-sensing technique is proposed. The contribution of this work is summarized as follows:

- Since the proposed scheme uses wireless signals from mobile devices of survivors, our proposal is relatively unaffected by obstacles. Using an UAV that can freely fly around the disaster area, rescuers can actively identify survivors, and the number of survivors can be managed in real time.
- To confirm the necessity of developing an asynchronous stream-sensing scheme for the case where an UAV uses Wi-Fi technology for a simple survivor discovery protocol, it is shown that asynchronous transmission naturally occurs depending on the path loss environment, the transmit power of the UAV, and the altitude of the UAV.
- We propose a deep-learning-based asynchronous stream-sensing scheme, which is developed by combining different types of neural networks such as a stacked autoencoder network, an inception layer-based network, and a linear network.
- To train our network, a framework is proposed, which connects a MATLAB-based simulator with a PyTorch-based network training code through the MATLAB-python API. Through this framework, our proposed network is trained via MATLAB by generating asynchronously transmitted signals generated in various situations in real time. Moreover, the number of possible cases of wireless channel environments

is almost infinite, and hence there is a limit to collecting training data in advance. Without the limitation, our framework allows the neural network to be continuously trained with various training data.

- Finally, extensive evaluation has proven that the proposed technique performs well enough to be used in a disaster situation. It also shows that the performance is superior to the existing mathematical techniques.

2. Related Works

Since its advent, deep learning has made remarkable progress in the theory and practice of many fields of science and technology. In [5,6], deep learning was especially powerful at extracting meaningful features from complex and diverse data that are difficult to analyze. This strength has been demonstrated by achieving astonishing results in computer vision [7], natural language processing [8], and audio-signal processing [9].

In recent years, several researchers have applied deep-learning technology to wireless communication fields such as physical layer [10], modulation classification [11,12], channel estimation [13], and decoding [14]. In addition, due to their powerful performance in classification, some researchers have applied machine-learning neural networks to spectrum sensing [15–20]. To decide whether a channel is occupied by a primary user, the work [15] adopted a basic neural network that stacks several linear layers. This method used the energy values and likelihood ratio statistics as input. In [16], a convolution neural network (CNN) was used using cyclostationary features and energy features as input. In [17], the authors also exploited a CNN for spectrum sensing in cognitive radio. Unlike [16], they considered covariance matrices of received signals as input. For the technique based on orthogonal frequency division multiplexing (OFDM), the researchers in [18] adopted a stacked autoencoder with basic linear layers. The proposed network used raw time-domain signals as input, and then the extracted features were passed to the classifier using logistic regression. To develop a context-aware cognitive radio, the authors in [19] utilized the power of each fast Fourier transform bin as input of their neural network. They verified the performance of machine-learning-based approaches using the software-defined radio platform. The authors in [20] proposed an unsupervised deep-learning-based spectrum sensing algorithm, while all the previous works in [15–19] proposed supervised learning-based spectrum sensing algorithms. The aforementioned studies confirmed that the powerful classification ability of neural networks can improve spectrum sensing performance, and this improvement proved that neural networks can extract essential features from signal samples.

The task of detecting the number of transmitted signals is more complicated than spectrum sensing that detects the presence or absence of signals. There have been several works that detected the number of signals [21,22]. The work [21] proposed two most common methods developed based on two famous information theoretic criteria: the Akaike information criterion (AIC) and the minimum description length (MDL) criterion [23–26]. The performance of the MDL estimator was analyzed asymptotically in [22]. If there are several observed received signal vectors, the optimal estimation on the number of sources based on the maximum likelihood probability can be achieved by comparing the sample covariance matrix obtained from the observed signal vectors and the ideal covariance matrix. These methods may be utilized in commercial OFDM-based wireless technologies such as Wi-Fi and LTE. However, when such detection schemes are adopted in OFDM-based communications, there is an inherent limitation that all transmitted signals should be synchronized from a cyclic prefix point of view. Therefore, these schemes will not work in the asynchronous transmission situation where the cyclic prefixes of the transmitted signals are not properly aligned. In other words, since synchronization between all transmitted signals should be required in existing OFDM-based communications, there is no work to develop a method for detecting asynchronously transmitted signals.

For a public safety network, 3GPP has standardized public safety-LTE (PS-LTE) [27]. PS-LTE is a technology that accommodates functions required for public safety communi-

cation based on existing LTE technology. This supports functions such as group communication, device-to-device direct communication (D2D), proximity service, and network survivability. However, these functions are primarily developed for situations where there is surviving network infrastructure in a disaster area. Considering the variety of disaster situations, it is necessary to develop a technology for scenarios where the communication infrastructure has been completely destroyed.

In addition, deep-learning technology is applied to the field of UAV/drone detection algorithms [28,29]. The work [28] developed a method to classify the types of drones, called drone classification. For the scheme, the authors utilized millimeter wave (mmWave) and a deep-learning network, which presents high drone-classification accuracy. The authors of [29] proposed a deep-learning-based framework to detect malicious drones, which outperforms existing state-of-the-art drone detection methods. However, our work is different from drone-detection research. Using wireless signals, our proposed scheme recognizes the number of devices around an UAV. Particularly, our novel deep-learning-based scheme can detect the number of asynchronous wireless signals.

3. Preliminary

This section describes the definitions of synchronously transmitted signals and asynchronously transmitted signals. Then, based on the limitations of UAVs, we explain why asynchronous transmission is better than synchronous transmission in the disaster situation that we considered in this work.

3.1. Synchronous and Asynchronous Transmissions

Figure 1a,b illustrate the examples of synchronous and asynchronous scenarios, respectively. For these examples, a transmitted signal is generated using OFDM, which is the most common modulation adopted in LTE and Wi-Fi [30,31]. Transmitted signals consist of several OFDM symbols, each of which is composed of a cyclic prefix (CP) part and a data part. In Figure 1a, all OFDM symbols are well aligned at the start of the cyclic prefix parts, representing the scenario of synchronous transmission. In contrast, Figure 1b illustrates the asynchronous transmission situation where all mobile devices start their transmissions randomly.

In general, if the CPs of two transmitted signals overlap with each other, it can be said that the two transmissions are synchronized. Figure 1c shows an example of overlapping CPs. In Figure 1c, when a receiver starts to extract time-domain samples from a received signal in the red region indicated by '(i)' where two CPs overlap, two signals in the extracted samples are synchronized due to the property of the CP. Therefore, the extracted samples in the blue region marked with '(ii)' have synchronized signals. On the other hand, as shown in Figure 1d, if there is no overlap of the CPs, this is called asynchronous transmission and synchronized signals cannot be extracted.

Figure 1e presents the case where synchronous and asynchronous transmission is mixed in a received signal. The extracted samples of Device 1 and Device 2 are synchronized, but those of Device 1 and Device 3 (or Device 4) are not synchronized. This means that the transmitted signals in a received signal can or cannot be synchronized with other transmitted signals. Therefore, the extracted samples from a received signal can be classified into three cases: (1) samples of synchronous transmission, (2) samples of asynchronous transmission, and (3) samples of a mixture of synchronous and asynchronous transmission.

In summary, perfectly synchronized signals cannot be obtained at a receiver if not all of the CPs of transmitted signals overlap. Furthermore, even if there is a region where all CPs overlap, synchronized signals cannot be obtained if a receiver does not start sampling within the overlapping region. Therefore, the input of our proposed network can be a mix of synchronous and asynchronous transmission and asynchronous transmission, as well as synchronous transmission.

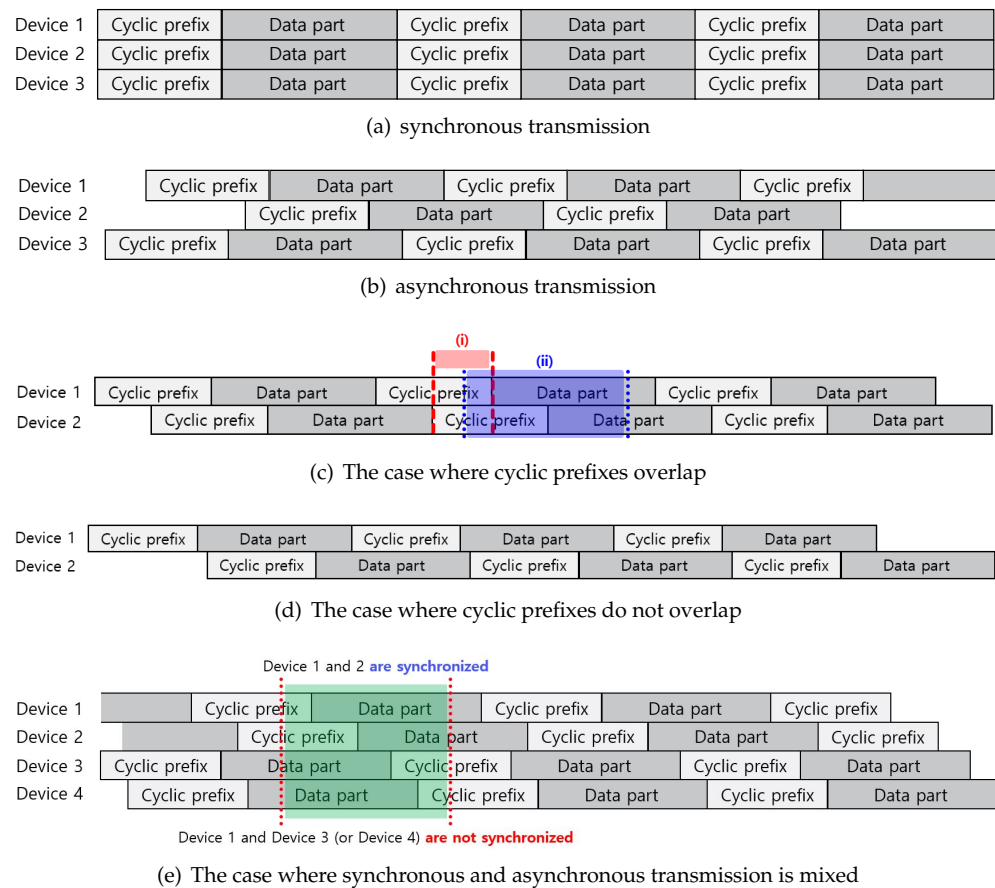


Figure 1. Example of CPs overlapping and not overlapping.

3.2. The Need for an Asynchronous Stream-Sensing Technique

In this subsection, we describe the need for an asynchronous stream-sensing technique by introducing the UAVs' limitations in synchronous transmission and by analyzing the asynchronicity in transmission over Wi-Fi depending on the placement of users and various pathloss models.

Limitations on the use of synchronous transmission for UAVs utilized for a survivor discovery purpose: To ensure synchronization between devices, a rescue UAV should serve as a base station in LTE that is a representative synchronous communication and utilizes a centralized synchronization process with several uplink channels [32,33]. That is, a rescue UAV should carry all the equipment required for the synchronization procedure. The equipment increases the overall load, which leads to an increase in the amount of power the UAV consumes for hovering and moving. UAVs, unfortunately, have battery limitations, and thus increasing power consumption will inevitably shorten the UAV's operating time. Moreover, the size of UAVs has been decreasing recently, which makes it infeasible to mount a lot of equipment on small UAVs.

Despite the importance of uptime in a disaster situation, due to the heavy equipment requirement, synchronous transmission can shorten the uptime of rescue UAVs that should search the entire disaster area. Moreover, with the variety of UAV sizes, it is necessary to develop a light survivor discovery technology that can be used regardless of the size of rescue UAVs.

Asynchronicity in Wi-Fi transmission: This work considers a scenario in which a rescue UAV and mobile devices transmit their signals using Wi-Fi, a lightweight wireless technology compared to LTE (Note that commercial UAVs such as the DJI Phantom series already use Wi-Fi).

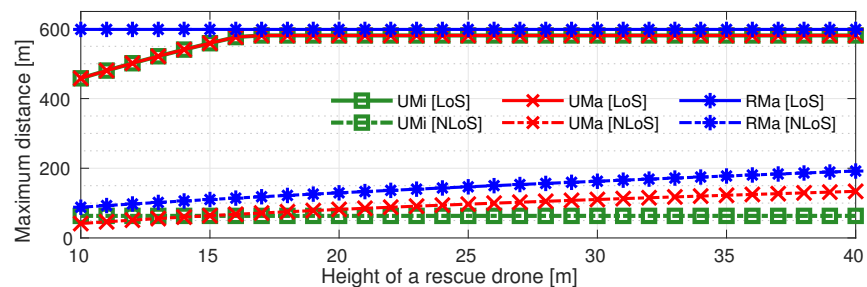
To determine the number of survivors, a rescue UAV sends a trigger message to collect rescue signals from mobile devices. As soon as mobile devices receive the message, they immediately begin sending rescue signals. The devices appear to be synchronized, but in reality they cannot be synchronized.

Depending on the distance between an UAV and each mobile device, the arrival time of the trigger message on each mobile device is different, and vice versa. If the difference between the arrival times of the trigger message in two mobile devices is T_{diff} , at the UAV, the difference between the arrival times of rescue signals is $2 \times T_{diff}$. That is, if $2 \times T_{diff} < T_{CP}$ is satisfied, where T_{CP} is the time duration of a CP, the CPs of two rescue signals can overlap. In Wi-Fi [34], T_{CP} can be set to 0.2 μs , 0.4 μs or 0.8 μs , and thus T_{diff} should be less than 0.1 μs , 0.2 μs , or 0.4 μs for the synchronization. In other words, the difference between the distances from two mobile devices to a rescue UAV must be less than $c \times T_{diff}$ where c is the speed of light. Hence, for each case, the maximum synchronization distance is 30 m, 60 m, or 120 m, respectively.

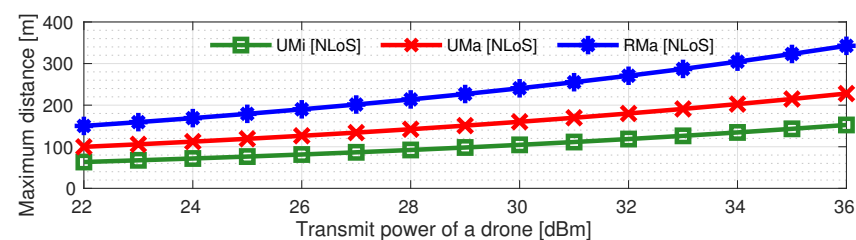
To investigate whether asynchronous transmission can occur, we calculated the maximum distance that a trigger message can be delivered to a mobile device considering various pathloss models: (1) Urban Micro (UMi), (2) Urban Macro (UMa), and (3) Rural Macro (RMa) [35]. The maximum distance is considered for both line of sight (LOS) and non-line-of-sight (NLOS). In this analysis, a trigger message is transmitted with the lowest modulation and coding scheme, the minimum receiver sensitivity of which is -82 dBm on Wi-Fi [31]. This analysis considers the transmission of a trigger message to be successful if the signal strength at the time of receiving the trigger message is greater than -82 dBm.

Figure 2a presents the maximum distance for various hovering heights of UAVs with a transmit power of 23 dBm. Under the LOS environment, the trigger message can be delivered to mobile devices over more than 400 m for all pathloss models. In almost all cases, the maximum distance is 600 m. That is, for all the path loss models, the maximum distance is larger than 30 m. For UMa and RMa even in NLOS, the maximum distance becomes greater than 120 m as the altitude of the UAV increases. Therefore, these results confirm that asynchronous transmission can frequently occur.

Figure 2b shows the results only under the NLOS environment for various transmission powers and an UAV altitude of 25 m. The transmission power ranges from 22 to 36 dBm for the outdoor scenario [36]. The maximum communication distance is larger than 30 m for all the pathloss models. Moreover, for UMa and RMa, the maximum communication distance is greater than 120 m in almost all cases.



(a) With various heights of a UAV for a transmit power of 23 dBm



(b) With various transmission power at a UAV height of 25m

Figure 2. The maximum communication distance for various pathloss models.

In summary, we can conclude that it is necessary to develop a generally available asynchronous stream-sensing technique. Through our analysis, even in the NLOS environment, asynchronous transmission can occur frequently depending on the characteristics of the disaster area, the altitude of the UAV, and the transmission power.

4. System Model

In Figure 3, our system model considers a disaster situation where all cell sites are destroyed by a large-scale natural disaster, and thus all survivors are unable to use the communication infrastructure. To quickly search for survivors, a rescue UAV equipped with N_R antennas is deployed in the disaster area. We assume that a mobile device with a single antenna broadcasts a short rescue signal. The short signal consumes little power, and thus allows survivors to use their device batteries as long as possible. Devices *asynchronously* transmit their rescue signals. Both the rescue UAV and the survivors' mobile devices utilize OFDM to receive and transmit signals. Each device transmits N_{sym} OFDM symbols as a rescue signal. In an OFDM symbol, the data part has N_{FFT} time-domain samples, which means that N_{FFT} subcarriers are used for transmission. As a result, since our survivor discovery system is based on OFDM and does not require any modification of the physical layer on Wi-Fi, we can maintain backward compatibility with Wi-Fi. (That is, our proposal just requires adding our survivor discovery protocol to the medium access control layer of Wi-Fi, which could be achieved with a simple update).

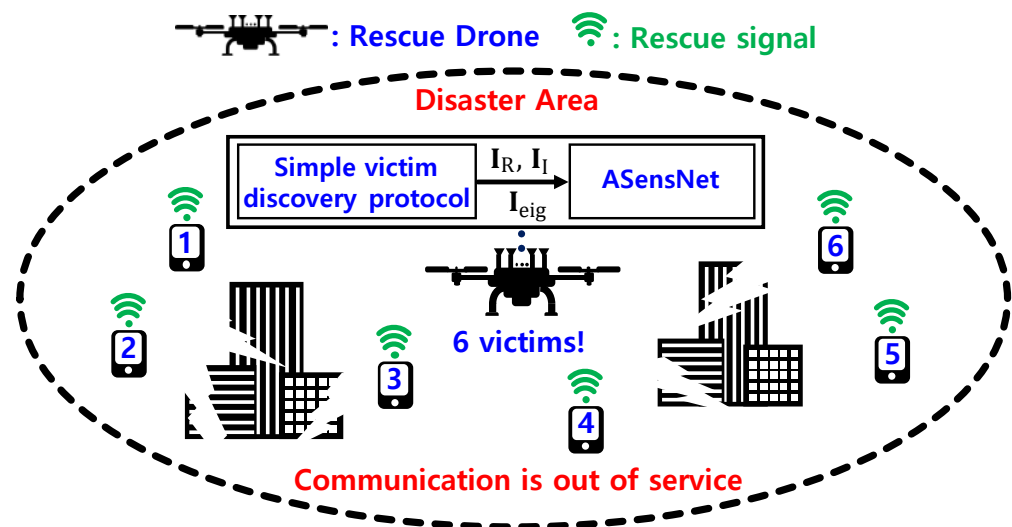


Figure 3. Disaster scenario in which all cell sites are destroyed by a natural disaster.

Figure 4 presents our simple survivor discovery protocol. To identify the number of mobile devices around itself, the rescue UAV broadcasts a trigger message. When mobile devices receive the message, they immediately begin to transmit rescue symbols. The UAV then collects rescue signals for $N_{\text{sym}} \cdot T_{\text{sym}}$, where T_{sym} is the duration of time of an OFDM symbol. The transmitted rescue signals sent simultaneously by mobile devices are superposed at the receiver. The rescue UAV executes our proposed asynchronous stream-sensing scheme on the received signal to determine how many mobile devices transmitted their rescue signals.

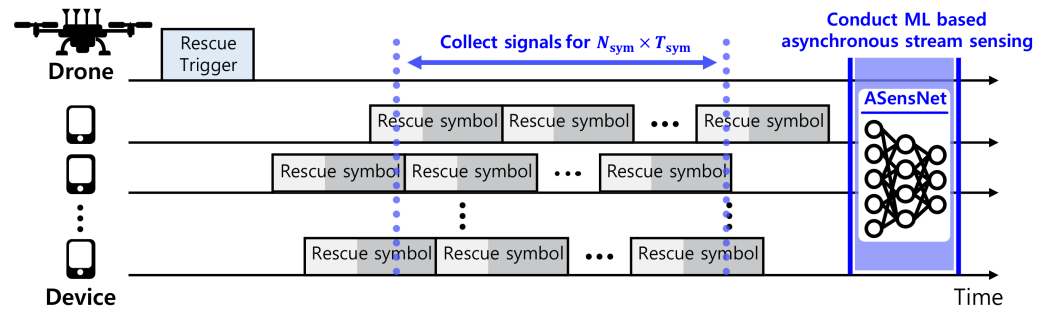


Figure 4. Survivor discovery protocol.

5. Learning-Based Asynchronous Stream Sensing

5.1. Input Data Generation

For our stream-sensing network, as in [17,20], we consider covariance matrices and eigenvalues of an expected covariance matrix. Then, the following describes in detail how the input data are generated.

For covariance matrices, we constructed four-dimensional (4D) matrices as input data. As mentioned in Section 4, a received signal for $N_{\text{sym}} \cdot T_{\text{sym}}$ includes multiple rescue signals that asynchronously overlap with each other. Since a rescue UAV is equipped with N_R antennas, it actually obtains N_R received signals. Under a general fading environment, N_R wireless channels from a mobile device's antenna to N_R rescue UAV's antennas are independent and all different, and thus, N_R received signals are also different from each other.

From N_R received signals in the time domain, we extract input data as follows. In time domain, for the r -th rescue UAV's antenna, an UAV receiver extracts $(N_{\text{CP}} + N_{\text{FFT}}) \times N_{\text{sym}}$ samples from a received signal. Note that the UAV receiver starts extracting time-domain samples from the moment it detects a signal, the energy of which exceeds a certain energy threshold. The extracted samples are divided into N_{sym} groups, each of which has $(N_{\text{CP}} + N_{\text{FFT}})$ samples. From each sample group, N_{FFT} samples are extracted from the $(N_{\text{CP}} + 1)$ -th sample to the $(N_{\text{CP}} + N_{\text{FFT}})$ -th sample. By using the inverse Fourier transform, the extracted time-domain samples are transformed to a frequency-domain samples. For the r -th rescue UAV's antenna, the extracted frequency-domain samples from the i -th sample group are denoted as $\mathbf{y}_{r,i} \in \mathbb{C}^{N_{\text{FFT}} \times 1}$. Based on $\mathbf{y}_{r,i}$, $r \in [1, N_R]$, $i \in [1, N_{\text{sym}}]$, we make $\mathbf{d}_{i,s} \in \mathbb{C}^{N_R \times 1}$ as

$$\mathbf{d}_{i,s} = [\mathbf{y}_{1,i}(s), \mathbf{y}_{2,i}(s), \dots, \mathbf{y}_{N_R,i}(s)]^T, s \in [1, N_{\text{FFT}}], \quad (1)$$

where $(\cdot)^T$ is the transpose operation, and $\mathbf{y}_{r,i}(s)$ is the s -th element of $\mathbf{y}_{r,i}$, which corresponds to a frequency-domain sample transmitted via the s -th subcarrier among N_{FFT} subcarriers. Hence, $\mathbf{d}_{i,s}$ is a vector that includes frequency domain samples transmitted via the s -th subcarrier in the i -th sample group for all rescue UAV's antennas. By using $\mathbf{d}_{i,s}$, we calculate a covariance matrix $\mathbf{D}_{i,s} \in \mathbb{C}^{N_R \times N_R} = \mathbf{d}_{i,s} \cdot (\mathbf{d}_{i,s})^H$ where $(\cdot)^H$ is Hermitian transpose. Finally, an input $\mathbf{I} \in \mathbb{C}^{N_R \times N_R \times N_{\text{FFT}} \times N_{\text{sym}}}$ is constructed as in Figure 5.

Note that by utilizing covariance matrices, the input data are similar to image data, and we can exploit existing neural networks for image classification.

Since neural networks cannot handle complex values, we split the input data into real input data and imaginary input data. Note that imaginary input data are obtained by calculating absolute values of imaginary values. In practice, the magnitude of received signals actually is very small ($\sim 10^{-7}$), and thus we additionally conduct normalization.

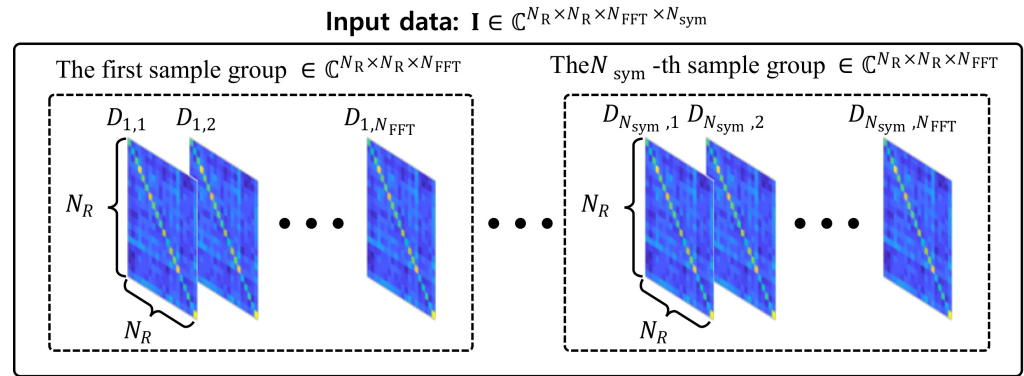


Figure 5. Example of input data.

Hence, real and imaginary input data are represented as follows:

$$\mathbf{I}_R = \frac{\Re(\mathbf{I})}{\max(\Re(\mathbf{I}))}, \mathbf{I}_I = \frac{\Im(\mathbf{I})}{\max(\Im(\mathbf{I}))}, \quad (2)$$

where the operators $\Re(\cdot)$ and $\Im(\cdot)$ extract real and imaginary values from \mathbf{I} , respectively. The operator $\max(\cdot)$ produces the maximum value in a matrix. Additionally, for \mathbf{I}_R and \mathbf{I}_I , the covariance matrices are redefined as $\mathbf{D}_{R,(i,s)}$ and $\mathbf{D}_{I,(i,s)}$, $i \in [1, N_{sym}]$, $s \in [1, N_{FFT}]$, respectively.

The input datum \mathbf{I} has N_{sym} sample groups, each of which has N_{FFT} covariance matrices. In a sample group, covariance matrices have the frequency domain information. Sample groups include information from the time-domain perspective. Therefore, the 4D input allows our network to simultaneously observe the correlation in terms of both time and frequency.

Eigenvalues of the input data are represented by $\mathbf{I}_{eig} \in \mathbb{R}^{N_{FFT} \cdot N_R \times 1}$ and generated by normalizing the eigenvalues of expected covariance matrices, $\mathbf{D}_{exp,s} = \sum_{i=1}^{N_{sym}} \mathbf{D}_{i,s}$, $\forall s$. Specifically, if $\mathbf{I}_{eig,s} \in \mathbb{R}^{N_R \times 1}$ is a vector that includes normalized eigenvalues of $\mathbf{D}_{exp,s}$, then $\mathbf{I}_{eig} = [(\mathbf{I}_{eig,1})^T, (\mathbf{I}_{eig,2})^T, \dots, (\mathbf{I}_{eig,N_{FFT}})^T]^T$.

5.2. Asynchronous Stream-Sensing Network

Figure 6 presents the overall structure of our asynchronous stream-sensing network (ASensNet). The ASensNet comprises five extractors: (1) an AutoEncoder Extractor, (2) an Inception-layer-based extractor, (3) linear extractor 1, (4) linear extractor 2, and (5) a Classifier. The AutoEncoder Extractor and the Inception-layer-based extractor have a real subnetwork and an imaginary subnetwork. Real subnetworks process \mathbf{I}_R , while imaginary subnetworks handle \mathbf{I}_I . For each extractor, the real and imaginary subnetworks have the same structure. With Linear Extractor 1, the extracted features from subnetworks merge into a feature. Finally, to detect the number of rescue signals in a received signal, the classifier deals with a combined feature of two features extracted from both linear extractor 1 and 2. Linear extractor 2 extracts features from \mathbf{I}_{eig} .

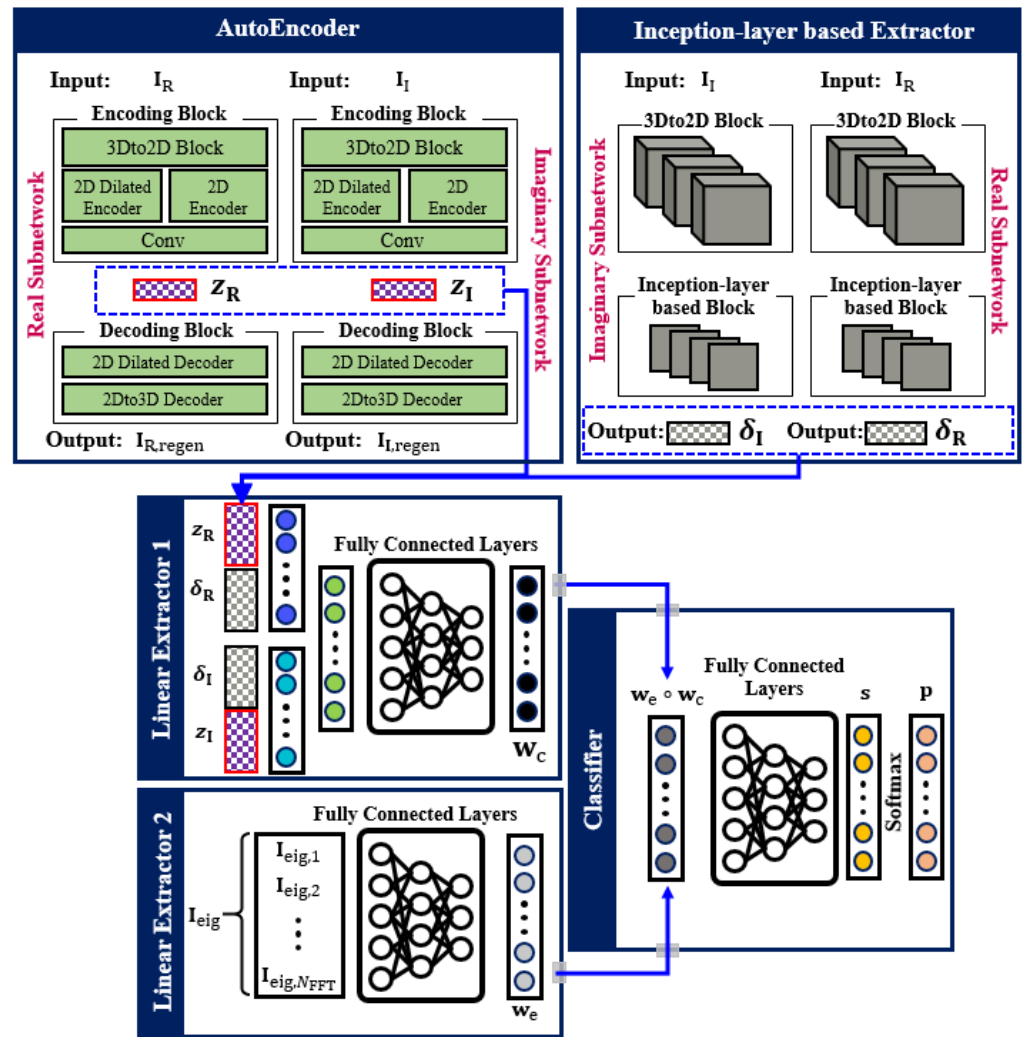


Figure 6. Asynchronous stream-sensing network (ASensNet).

5.2.1. Inception Layer-Based Extractor

This extractor consists of ‘Inception layer-based Block’ and ‘3Dto2D Block’.

Inception layer-based Block: The inception model is one of the most popular models for image classification and object detection in computer vision [37,38]. It is well known that the model is efficient due to its low computational load with a small number of model parameters. Hence, this model is appropriate for our ASensNet, which requires limited energy consumption due to the limited battery of UAVs.

For this block, we exploit an inception module and a grid reduction module in [38], and then stack both modules. Then, some dropout layers are inserted into the stacked network to prevent overfitting.

Finally, from ‘Inception layer-based Extractor’, subnetworks provide features δ_R and δ_I for I_R and I_I , respectively.

3Dto2D Block: This block is to shrink the dimension of I_R and I_I from 4D to three dimensions (3D) since two-dimensional (2D) convolution layers in the inception module can deal with 3D matrices. This block consists of some 3D convolution layers that handle 4D matrices. After I_R (or I_I) passes through 3D convolution layers, the result matrix is reshaped into a 3D matrix.

Through 3D convolution layers, this block is expected to capture temporal information from the input data. According to [39], 3D convolution layers model temporal information better than 2D convolution layers. The resulting matrix of this block can preserve temporal

information. Therefore, ‘3Dto2D Block’ not only reduces the dimension of input data, but also allows the inception layer-based extractor to take into account the temporal information of a received signal.

5.2.2. Autoencoder Extractor

In ASensNet, the AutoEncoder Extractor is based on the deep autoencoder that is widely used for efficient encoding and reduction of dimensionality [40]. Such a deep autoencoder structure is specialized to extract low-dimensional core features from high-dimensional data. Therefore, using the autoencoder structure, this extractor is responsible for extracting features from covariance matrices $\mathbf{D}_{i,s}, \forall i, s$ in \mathbf{I} . Specifically, for the autoencoder network, each subnetwork regenerates the input \mathbf{I}_R (or \mathbf{I}_I) in the output. For real and imaginary subnetworks, ‘Encoding Block’ extracts real and imaginary latent vectors denoted as \mathbf{z}_R and \mathbf{z}_I , respectively. ‘Decoding Block’ generates $\mathbf{I}_{R, \text{regen}}$ and $\mathbf{I}_{I, \text{regen}}$ based on \mathbf{z}_R and \mathbf{z}_I , respectively.

Feature Extraction: Since ‘Decoding Block’ aims to make $\mathbf{I}_{R, \text{regen}}$ and $\mathbf{I}_{I, \text{regen}}$ the same as \mathbf{I}_R and \mathbf{I}_I , ‘Encoding Block’ should generate \mathbf{z}_R and \mathbf{z}_I that include low-dimensional necessary information for ‘Decoding Block’ to regenerate the input. In other words, for subnetworks, the latent vectors \mathbf{z}_R and \mathbf{z}_I are bound to contain the most essential features that represent $\mathbf{D}_{R,(i,s)}$ and $\mathbf{D}_{I,(i,s)}$ in the input \mathbf{I}_R and \mathbf{I}_I , respectively.

As a result, from $\mathbf{D}_{R,(i,s)}$ and $\mathbf{D}_{I,(i,s)}$ in \mathbf{I}_R and \mathbf{I}_I , ‘Encoding Block’ extracts distinct low-dimensional features depending on the number of rescue signals in a received signal.

2D Dilated Encoder: In our AutoEncoder Extractor, there are two types of two-dimensional encoder: a ‘2D Encoder’ and a ‘2D Dilated Encoder’. The 2D dilated encoder is constructed by stacking 2D dilated convolution layers, while the 2D encoder consists of 2D convolution layers, similar to a typical convolutional neural network. By inserting zero between the values in a kernel, a 2D dilated convolution performs convolution with non-adjacent values in an input matrix, which provides a wider field of view at the same computational cost [41].

For this block, we attach a 2D dilated encoder. Due to the structural feature of the kernel of the 2D dilated convolution, the 2D dilated encoder considers the relationship among non-adjacent values of $\mathbf{D}_{R,(i,s)}$ (or $\mathbf{D}_{I,(i,s)}$). Since $\mathbf{D}_{i,s} = \mathbf{d}_{i,s} \cdot (\mathbf{d}_{i,s})^H$, considering between non-adjacent values in $\mathbf{D}_{R,(i,s)}$ (or $\mathbf{D}_{I,(i,s)}$) has the same effect as taking non-adjacent values of $\mathbf{d}_{i,s}$ into account (Note that the r -th element of $\mathbf{d}_{i,s}$ is the signal value obtained from the r -th receive antenna).

The matrix $\mathbf{D}_{R,(i,s)}$ is a covariance matrix of a vector $\mathbf{d}_{i,s}$ that is obtained from a received signal. The received signal consists of multiple signals transmitted from multiple devices and propagating over wireless channels. In the case, the independence of wireless channels can theoretically be preserved only when all transmitted signals are well synchronized. Otherwise, if signals are transmitted asynchronously from devices, the independence is no longer maintained (This is also why the existing mathematical schemes AIC and MDL cannot work with asynchronous transmission, because they were developed based on the assumption of synchronous transmission). The asynchronicity in a received signal could cause in $\mathbf{d}_{i,s}$ channel correlation among non-adjacent antennas. As a result, through ‘2D Dilated Encoder’ using 2D dilated convolution layers, our AutoEncoder Extractor produces \mathbf{z}_R and \mathbf{z}_I considering spatial correlation among non-adjacent antennas.

5.2.3. Linear Extractor 1

This extractor fetches four features, \mathbf{z}_R , \mathbf{z}_I , δ_R , and δ_I , from the AutoEncoder Extractor and the Inception-layer-based extractor. Then, \mathbf{z}_R and δ_R are concatenated, while \mathbf{z}_I and δ_I are attached. Each concatenated feature is passed through a 2D convolution layer. Both resultant feature vectors are concatenated into a single vector, and then several fully connected layers are followed. Then, the fully connected layers produce a vector \mathbf{w}_c .

5.2.4. Linear Extractor 2

Through some fully connected layers, this extractor obtains a feature vector from \mathbf{I}_{eig} , which is represented by \mathbf{w}_e .

5.2.5. Classifier

Using a vector obtained by the Hadamard product of \mathbf{w}_e and \mathbf{w}_c , it predicts a score vector \mathbf{s} . This score vector will be used to calculate the loss of this Classifier.

Prediction: the probability vector \mathbf{p} is calculated by passing the predicted score vector \mathbf{s} through a softmax layer.

5.3. Losses and Objectives for Learning ASensNet

For training samples, \mathcal{I}_R and \mathcal{I}_I denote input domains for \mathbf{I}_R and \mathbf{I}_I , respectively. Samples of \mathcal{I}_R and \mathcal{I}_I are followed by a data distribution, denoted as $\mathbf{I}_R \sim p_{\text{data}}(\mathbf{I}_R)$ and $\mathbf{I}_I \sim p_{\text{data}}(\mathbf{I}_I)$, respectively. The input domain of \mathbf{I}_{eig} is represented by \mathcal{I}_{eig} , and samples are followed by $\mathbf{I}_{\text{eig}} \sim p_{\text{data}}(\mathbf{I}_{\text{eig}})$. The label domain is represented by \mathcal{L} . Feature domains of $\mathbf{z}_R, \mathbf{z}_I, \delta_R$, and δ_I are represented as $\mathcal{Z}_R, \mathcal{Z}_I, \Delta_R$, and Δ_I , respectively. In addition, \mathcal{F} denotes a domain constructed by all feature vectors $\mathbf{z}_R, \mathbf{z}_I, \delta_R$, and δ_I . The domains for $\mathbf{w}_e, \mathbf{w}_c$, and \mathbf{s} are represented by $\mathcal{W}_e, \mathcal{W}_c$, and \mathcal{S} , respectively. The domain for $(\mathbf{w}_e \circ \mathbf{w}_c)$ is denoted as \mathcal{W}_{eoc} . Finally, $O(\cdot)$ represents a loss function, and then this work uses the support vector machine (SVM) for multi-class classification [42]. This SVM loss function takes as input a score vector and a ground truth.

For our ASensNet framework, we have four subnetworks for extracting features from input data and a fully connected-based classifier. For two subnetworks of the AutoEncoder Extractor, two Encoding Blocks can be represented as mapping functions: $E_R: \mathcal{I}_R \rightarrow \mathcal{Z}_R$ and $E_I: \mathcal{I}_I \rightarrow \mathcal{Z}_I$, respectively, while two Decoding Blocks are also two mapping functions: $D_R: \mathcal{Z}_R \rightarrow \mathcal{I}_R$ and $D_I: \mathcal{Z}_I \rightarrow \mathcal{I}_I$, respectively. For the inception layer-based extractor, the real and imaginary subnetworks are also mapping functions: $F_R: \mathcal{I}_R \rightarrow \Delta_R$ and $F_I: \mathcal{I}_I \rightarrow \Delta_I$. The Linear extractor 1 and 2 are represented as: $K_c: \mathcal{F} \rightarrow \mathcal{W}_c$ and $K_e: \mathcal{I}_{\text{eig}} \rightarrow \mathcal{W}_e$, respectively. Finally, the Classifier is a mapping function: $K: \mathcal{W}_{\text{eoc}} \rightarrow \mathcal{S}$.

5.3.1. Losses for AutoEncoder Extractor

The losses of the AutoEncoder Extractor are defined for each ‘Encoding Block’ and ‘Decoding Block’.

Losses for Encoding Block: The Encoding Blocks should make the Decoding Blocks generate the same $\mathbf{I}_{R,\text{regen}}$ and $\mathbf{I}_{I,\text{regen}}$ as \mathbf{I}_R and \mathbf{I}_I as much as possible. Therefore, for real and imaginary subnetworks, the regeneration losses are defined as

$$\mathcal{L}_{\text{regen}, R}(E_R, D_R, \mathcal{I}_R) = \mathbb{E}_{\mathbf{I}_R} [D_R(E_R(\mathbf{I}_R)) - \mathbf{I}_R], \quad (3)$$

$$\mathcal{L}_{\text{regen}, I}(E_I, D_I, \mathcal{I}_I) = \mathbb{E}_{\mathbf{I}_I} [D_I(E_I(\mathbf{I}_I)) - \mathbf{I}_I]. \quad (4)$$

In addition, the Encoding Blocks should also generate \mathbf{z}_R and \mathbf{z}_I that allow the Classifier to predict \mathbf{p} as well as possible. The classification loss is defined as (5).

$$\mathcal{L}_{\text{class}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L}) = \mathbb{E}_{\mathbf{I}_R, \mathbf{I}_I, \mathbf{I}_{\text{eig}}} [O(K(\mathbf{w}_e \circ \mathbf{w}_c), G_{\text{truth}})], \quad (5)$$

where $\mathbf{w}_c = K_c(E_R(\mathbf{I}_R), E_I(\mathbf{I}_I), F_R(\mathbf{I}_R), F_I(\mathbf{I}_I))$ and $\mathbf{w}_e = K_e(\mathbf{I}_{\text{eig}})$. In addition, G_{truth} is given as soon as $\mathbf{I}_R, \mathbf{I}_I$, and \mathbf{I}_{eig} are generated. Note that the loss is the same for the real and imaginary subnetworks.

Objective of AutoEncoder Extractor: Therefore, the complete objective of the AutoEncoder Extractor is defined as

$$\begin{aligned} & \mathcal{L}_{\text{AE}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L}) \\ &= \mathcal{L}_{\text{regen,R}}(E_R, D_R, \mathcal{I}_R) + \mathcal{L}_{\text{regen,I}}(E_I, D_I, \mathcal{I}_I) + \mathcal{L}_{\text{class}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L}). \end{aligned} \quad (6)$$

Note that the losses of the Decoding Blocks are the same as the losses of the Encoding Blocks. In other words, by $\mathcal{L}_{\text{regen,R}}$ and $\mathcal{L}_{\text{regen,I}}$, the Encoding Blocks and the Decoding Blocks are updated simultaneously. In the same vein, $\mathcal{L}_{\text{class}}$ updates both Encoding Blocks in the real and imaginary subnetworks.

For the AutoEncoder Extractor, the following problem will be solved.

$$E_R^*, E_I^* = \underset{E_R, E_I}{\operatorname{argmin}} \mathcal{L}_{\text{AE}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L}). \quad (7)$$

5.3.2. Losses for the Inception Layer-Based Extractor, Linear Extractors, and Classifier

The inception layer-based extractor, and linear extractor 1 and 2 should make the Classifier predict the probability vector \mathbf{p} as accurately as possible. Therefore, these extractors use the same classification loss as $\mathcal{L}_{\text{class}}$ to update their networks. In addition, the loss of the classifier is naturally defined as $\mathcal{L}_{\text{class}}$.

Objective of the Inception layer-based Extractor, Linear Extractors, and Classifier: The objective of these extractors is defined as $\mathcal{L}_{\text{class}}$. As a result, we can find F_R, F_I, K_c, K_e , and K by solving the following problem.

$$F_R^*, F_I^*, K_c^*, K_e^*, K^* = \underset{F_R, F_I, K_c, K_e, K}{\operatorname{argmin}} \mathcal{L}_{\text{class}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L}). \quad (8)$$

5.3.3. Loss for Updating ‘3Dto2D Block’ of F_R and F_I

The ‘3Dto2D Block’ of F_R and F_I is represented by B_R and B_I , respectively. The feature domain formed by concatenating the output from B_R and B_I is defined as \mathcal{M} . An auxiliary classifier is defined as $K_{\mathcal{M}}$, a mapping function: $K_{\mathcal{M}} : \mathcal{M} \rightarrow \mathfrak{S}$.

The loss function for $K_{\mathcal{M}}$ is defined as

$$\mathcal{L}_{K_{\mathcal{M}}}(B_R, B_I, K_{\mathcal{M}}, \mathcal{I}_R, \mathcal{I}_I, \mathcal{L}) = \mathbb{E}_{\mathbf{I}_R, \mathbf{I}_I} \left[O \left(K_{\mathcal{M}}(M_{\text{concat}}(B_R(\mathbf{I}_R), B_I(\mathbf{I}_I))), G_{\text{truth}} \right) \right], \quad (9)$$

where M_{concat} concatenates the output from B_R and B_I .

To update B_R and B_I , the following problem should be solved.

$$B_R^*, B_I^* = \underset{B_R, B_I}{\operatorname{argmin}} \mathcal{L}_{K_{\mathcal{M}}}(B_R, B_I, K_{\mathcal{M}}, \mathcal{I}_R, \mathcal{I}_I, \mathcal{L}). \quad (10)$$

The auxiliary classifier mitigates the vanishing gradient problem [37]. Through $\mathcal{L}_{K_{\mathcal{M}}}$, from the initial stage, it is expected that the features of the input data will be better categorized according to the number of transmitted signals. Note that the auxiliary classifier is used only for training, not for prediction.

5.4. Training Methodology

5.4.1. Training Procedure

The overall procedure is summarized in Algorithm 1, where N_{data} is the number of data. In this work, N_{data} is batch size. The feature extractors and the classifier are updated sequentially.

Algorithm 1 Training Procedure for One Epoch

Output: $E_R^*, E_I^*, F_R^*, F_I^*, K_c^*, K_e^*$ and K^*
for $i = 1 : N_{\text{data}}$
 (1) Obtain I_R, I_I , and I_{eig}
 (2) Update E_R, E_I with F_R, F_I, K_c, K_e, K
 Equation (6): $\underset{E_R, E_I}{\text{argmin}} \mathcal{L}_{\text{AE}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L})$
 (3) Update B_R, B_I with K_M
 Equation (9): $\underset{B_R, B_I}{\text{argmin}} \mathcal{L}_{K_M}(B_R, B_I, K_M, \mathcal{I}_R, \mathcal{I}_I, \mathcal{L})$
 (4) Update F_R, F_I, K_c, K_e, K with E_R, E_I
 Equation (8): $\underset{F_R, F_I, K_c, K_e, K}{\text{argmin}} \mathcal{L}_{\text{class}}(E_R, E_I, F_R, F_I, K_c, K_e, K, \mathcal{I}_R, \mathcal{I}_I, \mathcal{I}_{\text{eig}}, \mathcal{L})$
end for

5.4.2. Training Framework

To train our ASensNet, we developed a training framework that combines MATLAB and PyTorch. The input data I_R and I_I vary according to the wireless channel between a rescue UAV and mobile devices. In practice, the wireless channel is changed quickly and is infinitely diverse. Hence, it is impossible to collect adequately correlated dataset for all channel cases in advance. In other words, it is not easy to create an unbiased channel dataset. For this work, it is inappropriate to use the typical way of learning with a precollected dataset.

Algorithm 2 shows our training architecture. Our training framework generates input data in real time, and then updates our ASensNet. The MATLAB-based generator generates N_{data} received signals by asynchronously transmitting rescue signals. Using the MATLAB Engine API for Python [43], the generated data are sent to a PyTorch-based learning function. Then, the PyTorch-based learning function updates the ASensNet with the generated data. As a result, with our framework, the ASensNet is continuously updated with various input data. Note that generated channels are realistic by considering large-scale fading, small-scale fading with multiple channel taps, and various mobile device distributions. The realistic elements create an infinite number of channel cases. Therefore, this training approach does not incur an overfitting problem. The problem occurs when a deep-learning network is trained with a limited amount of data. That is, the deep learning network is biased towards the given data. However, our training approach is to train the ASensNet with infinitely changing channel data. In other words, the proposed network can be trained with an unbiased channel dataset.

The MATLAB-based generator randomly deploys N_{device} mobile devices around a rescue UAV. The UAV is equipped with ten antennas ($N_R = 10$), while mobile devices have a single antenna. Mobile devices start their transmission asynchronously. For the outdoor wireless channel between the UAV and mobile devices, the ITU RMa pathloss model [35] is implemented as large-scale fading, and five multipath taps are used for small-scale fading. The transmit power of an UAV and a device is set to 36 dBm, complying with the regulation [36]. Finally, the altitude of the UAV is assumed to be 25 m. For a channel impulse response, each channel tap from the first to last one exponentially decreases in magnitude. For time-varying channels, all channel impulse responses change every time N_{device} is determined. For time-varying channels, all channel impulse responses change whenever N_{device} is changed. With various $N_{\text{device}} \in [0, 9]$, the deployment of mobile devices changes with the change in impulse responses. Mobile devices are randomly deployed around a rescue UAV, and the SNRs of the signals transmitted from the mobile devices are different. Note that at the UAV, the difference in arrival time between the earliest signal and each of the other signals is actually determined randomly within $[0, T_{\text{sym}})$. In reality, the SNR value is not necessarily proportional to the distance between the receiver and transmitter. Even if the distance is short, the SNR value may be small depending on the surrounding environment of both the transmitter and receiver. Hence, the difference in arrival time of the signals is randomly selected regardless of SNR values, from which we

can believe that our evaluation can cover all possible situations in the real world. Finally, $N_{\text{FFT}} = 64$, $N_{\text{sym}} = 20$, and $T_{\text{CP}} = 0.2 \mu\text{sec}$.

Algorithm 2 Training Architecture

Initialization

- (1) Start running MATLAB and Python.
- (2) Determine N_{data} .
- (3) Go to ‘MATLAB-based Generator’.

procedure MATLAB-BASED GENERATOR

- (1) Start collecting N_{data} input data.
 - for** $k \leftarrow 1$ to N_{data} **do**
 - Select N_{device} within $[0, N_R - 1]$.
 - Drop N_{device} mobile devices randomly around a rescue drone.
 - Generate a received signal by the mobile devices.
 - If $N_{\text{device}} = 0$, generate a noise signal.
 - Make the k -th data, $\mathbf{I}_{R,k}$ and $\mathbf{I}_{I,k}$.
 - end for**
- (2) Return all $\mathbf{I}_{R,k}$ and $\mathbf{I}_{I,k}$, $\forall k$ to ‘PyTorch-based Learning Function’.

end procedure

procedure PYTORCH-BASED LEARNING FUNCTION

- (1) By Algorithm 1, update ASensNet with $\mathbf{I}_{R,k}$ and $\mathbf{I}_{I,k}$, $\forall k$.
- (2) To obtain input data, call ‘MATLAB-based Generator’ through MATLAB-Python API.

end procedure

Note that the majority of the literature on the development of machine-learning-based wireless communication and signal processing schemes also verified their proposed schemes through simulations using realistic channel generation [44–51]. Finally, the training details are summarized in Table 1.

Table 1. Model complexity and hyperparameters.

Parameters	Values
Mini batch size	10
Initial learning rate	1×10^{-3}
Scheduler	CosineAnnealingLR
Optimizer	RMSprop
Trainable parameters	260 M
Model Parameter size	1032 M

6. Evaluation

This section evaluates the ASensNet, which is compared to the existing mathematical schemes in [21]. The compared schemes are the Akaike information criterion (AIC) and minimum description length (MDL). Note that both schemes are meant for a scenario in which a received signal is constructed by synchronously transmitted signals. Since there is no work on asynchronous stream sensing, those two schemes are sufficient as comparison schemes. In the synchronous scenario, the precision and recall of AIC and MDL are greater than 0.9.

We have evaluated our ASensNet, AIC, and MDL for two scenarios. The first scenario is that transmitted signals of all mobile devices have almost the same SNR value at the receiver of a rescue UAV. Through evaluation, we investigate the performance of those schemes for each SNR value. The second scenario is that the transmitted signals of all mobile devices have different SNR values. This is more like a real-world scenario, from which we can assess how well the ASensNet will work in reality.

For the two comparison schemes, the maximum detectable number of signals depends on N_R . Specifically, both can detect up to $(N_R - 1)$ transmitted signals. Therefore, ASensNet, AIC, and MDL are compared for 0 to $(N_R - 1)$ transmitted signals. (In theory, for the synchronous scenario, a receiver equipped with N_R antennas can detect up to $(N_R - 1)$ transmitted signals from a received signal. Since this is due to a theoretical limitation of mathematical frameworks MDL and AIC, the maximum number of devices is set to $(N_R - 1)$ in this work.)

6.1. Performance Metrics

For performance evaluation in the rescue scenario, the following relaxed performance metrics are introduced:

- Precision with d for k transmitted signals ($P_{k\pm d}$): the relaxed precision is defined as follows.

$$P_{k,d} = \frac{TP_{k\pm d}}{TP_{k\pm d} + FP_{k\pm d}}, \quad (11)$$

where $TP_{k\pm d}$ and $FP_{k\pm d}$ are true positive and false positive, respectively. The parameter $TP_{k\pm d}$ denotes the case when a scheme predicts k transmitted signals; the actual number of transmitted signals is within $[k - d, k + d]$. The parameters $FP_{k\pm d}$ represent the case that when the detected number of transmitted signals is k , the actual number of transmitted signals is not within $[k - d, k + d]$.

- Recall with d for k transmitted signals ($R_{k\pm d}$): the relaxed precision is defined as follows.

$$R_{k,d} = \frac{TP_{k\pm d}}{TP_{k\pm d} + FN_{k\pm d}}, \quad (12)$$

where $FN_{k\pm d}$ is false negative and includes the case that when the actual number of transmitted signals is k , the detected number of transmitted signals is not within $[k - d, k + d]$.

Table 2 shows how to set $TP_{k\pm d}$, $FP_{k\pm d}$, and $FN_{k\pm d}$ for $k = 2, d = 1$.

The recall $R_{k,d}$ is within $[0, 1]$. As $R_{k,d}$ becomes closer to 1, the detection performance of a scheme becomes better. In general, $R_{k,d}$ is an index indicating how well the detected value is within $[k - d, k + d]$ when the actual number of transmitted signals is k . Hence, the higher the recall value, the better the detection performance of a detection scheme.

The precision $P_{k,d}$ is also within $[0, 1]$. As $P_{k,d}$ becomes closer to 1, the reliability of a scheme becomes better. For example, if $P_{2,1} = 1$, it means that for the predicted $k = 2$ transmitted signals, and then the actual number of transmitted signals is within $[1, 3]$ with probability of 1. On the other hand, for $P_{2,1} = 0$, although the detected number is $k = 2$, the actual number of transmitted signals is not within $[1, 3]$ with a probability of 0. As a result, the precision is an indicator of how reliable the predicted results are.

For this work, the reason why we have introduced relaxed performance metrics for the performance evaluation can be explained as follows. In a rescue situation, it might be more important to provide a rough overview of the situation without critical errors. In other words, even if the predicted value is slightly inaccurate, it can still be considered meaningful enough as the predicted value is reasonably close to the actual value from a contextual point of view. Note that for $k \in [0, 2]$, we show the precision and recall with only $d = 0$ since the error of 1 is relatively large compared to the actual values 0, 1, and 2. Furthermore, in computer vision, relaxed performance metrics are also used as the standard recall-at-k metric to evaluate how well a proposed system predicts values as close as possible to the true values [52–54].

In this paper, by comparing those results with $d \in \{0, 1\}$, we verify that our scheme with only the minimal relaxation value ($d = 1$) provides a reliable performance in various SNR scenarios while outperforming the existing solutions, AIC and MDL. Based on the results, our ML-based stream-sensing scheme shows high precision and recall performance

Table 4. Precision and recall of AIC.

Precision		The Number of Transmitted Signals									
SNR	d	0	1	2	3	4	5	6	7	8	9
6 dB	0	1.00	1.00	0.27	0.00	0.00	0.04	0.47	0.58	0.26	0.20
	1	-	-	-	0.11	0.00	0.09	0.72	0.89	0.49	0.29
8 dB	0	1.00	1.00	0.13	0.00	0.00	0.03	0.52	0.57	0.26	0.21
	1	-	-	-	0.13	0.00	0.08	0.73	0.88	0.48	0.29
10 dB	0	1.00	1.00	0.20	0.06	0.00	0.03	0.54	0.57	0.26	0.20
	1	-	-	-	0.11	0.00	0.07	0.74	0.89	0.50	0.29
12 dB	0	1.00	1.00	0.04	0.00	0.00	0.04	0.50	0.53	0.26	0.20
	1	-	-	-	0.09	0.00	0.08	0.73	0.89	0.49	0.29
14 dB	0	1.00	1.00	0.10	0.00	0.00	0.03	0.50	0.57	0.26	0.20
	1	-	-	-	0.00	0.00	0.08	0.72	0.88	0.49	0.29
Recall		The Number of Transmitted Signals									
SNR	d	0	1	2	3	4	5	6	7	8	9
6 dB	0	1.00	0.03	0.00	0.00	0.00	0.02	0.13	0.36	0.59	0.99
	1	-	-	-	0.00	0.00	0.04	0.31	0.70	1.00	1.00
8 dB	0	1.00	0.03	0.00	0.00	0.00	0.01	0.15	0.36	0.61	1.00
	1	-	-	-	0.00	0.00	0.03	0.31	0.73	1.00	1.00
10 dB	0	1.00	0.03	0.00	0.00	0.00	0.01	0.16	0.35	0.57	1.00
	1	-	-	-	0.00	0.00	0.03	0.34	0.70	1.00	1.00
12 dB	0	1.00	0.04	0.00	0.00	0.00	0.02	0.15	0.33	0.59	0.99
	1	-	-	-	0.00	0.00	0.03	0.35	0.69	1.00	1.00
14 dB	0	1.000	0.03	0.00	0.00	0.00	0.01	0.15	0.36	0.58	0.99
	1	-	-	-	0.00	0.00	0.03	0.32	0.71	1.00	1.00

Table 5. Precision and recall of MDL.

Precision		The Number of Transmitted Signals									
SNR	d	0	1	2	3	4	5	6	7	8	9
6 dB	0	1.00	0.91	0.33	0.00	0.00	0.00	0.05	0.30	0.22	0.14
	1	-	-	-	0.10	0.00	0.01	0.27	0.67	0.51	0.35
8 dB	0	1.00	1.00	0.00	0.00	0.00	0.00	0.11	0.30	0.20	0.14
	1	-	-	-	0.20	0.00	0.01	0.30	0.68	0.52	0.36
10 dB	0	1.00	1.00	0.06	0.00	0.00	0.00	0.10	0.31	0.21	0.15
	1	-	-	-	0.30	0.00	0.00	0.32	0.73	0.51	0.36
12 dB	0	1.00	0.92	0.09	0.00	0.00	0.00	0.08	0.31	0.20	0.15
	1	-	-	-	0.07	0.00	0.02	0.30	0.69	0.51	0.36
14 dB	0	1.00	1.00	0.00	0.00	0.00	0.00	0.07	0.34	0.20	0.14
	1	-	-	-	0.15	0.00	0.01	0.27	0.71	0.50	0.36

Table 5. Cont.

Recall		The Number of Transmitted Signals									
SNR	d	0	1	2	3	4	5	6	7	8	9
6 dB	0	1.00	0.01	0.00	0.00	0.00	0.00	0.01	0.13	0.41	1.00
	1	-	-	-	0.00	0.00	0.00	0.08	0.43	1.00	1.00
8 dB	0	1.00	0.01	0.00	0.00	0.00	0.00	0.03	0.13	0.36	1.00
	1	-	-	-	0.00	0.00	0.00	0.10	0.45	1.00	1.00
10 dB	0	1.00	0.01	0.00	0.00	0.00	0.00	0.02	0.12	0.41	1.00
	1	-	-	-	0.00	0.00	0.00	0.09	0.49	1.00	1.00
12 dB	0	1.00	0.01	0.00	0.00	0.00	0.00	0.02	0.13	0.40	1.00
	1	-	-	-	0.00	0.00	0.01	0.10	0.47	1.00	1.00
14 dB	0	1.00	0.01	0.00	0.00	0.00	0.00	0.02	0.14	0.40	1.00
	1	-	-	-	0.00	0.00	0.00	0.08	0.47	1.00	1.00

In Table 3, when $d = 0$, for all SNR values, precision and recall values decrease as the number of transmitted signals increases. This naturally means that it is difficult to detect a large number of transmitted signals. From $k = 6$, precision values become smaller than 0.9, while recall values become smaller than 0.9 from $k = 7$.

Moreover, our proposed scheme has high reliability with respect to the least relaxed criterion of $d = 1$. The precision values for $d = 1$ are higher than 0.9 in all cases. In other words, if a value detected by our ASensNet is k , the actual number of signals is within $[k - 1, k + 1]$, with probability greater than 0.9. As in precision, the recall values for $d = 1$ are larger than 0.9 in all cases. As a result, while our scheme shows high detection performance in most cases of the number of transmitted signals at $d = 0$, it also has high detection performance in terms of the minimum relaxation criterion of $d = 1$.

In Tables 4 and 5, both AIC and MDL have small precision and recall in almost all cases. Although existing solutions have large recall values for $k = 8$ and $k = 9$, the precision is very low. Only for $k = 0$, both schemes show high precision and recall. Even with respect to the relaxed criterion $d = 1$, the values detected by these techniques are not reliable. As a result, they can only be used to determine whether there is a transmitted signal or not.

To understand in detail why precision and recall become larger as d increases from 0 to 1, we investigate the confusion maps of our ASensNet, AIC, and MDL. Figure 7 shows the confusion maps of three schemes for the SNR of 6dB. For each scheme, the tendency of confusion maps for other SNR values is the same as that of Figure 7.

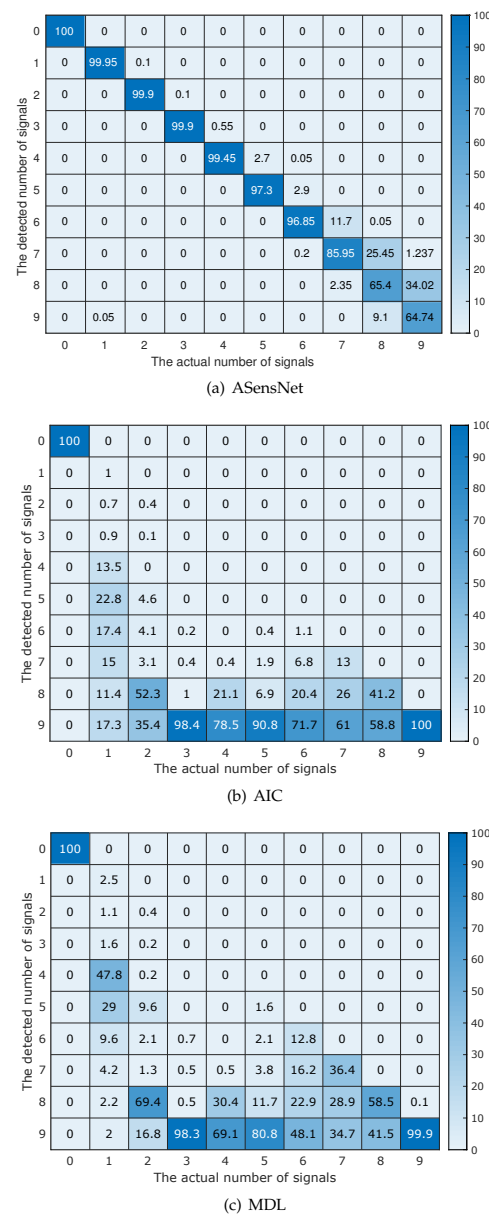


Figure 7. Confusion maps of ASensNet, AIC, and MDL for the SNR of 6dB.

Figure 7a shows that our ASensNet usually misjudges k transmitted signals as $(k + 1)$ or $(k - 1)$ transmitted signals. Therefore, for our proposed scheme, the precision and recall with $d = 1$ are greater than 0.9 in all cases.

Figure 7b,c, however, show that regardless of the number of signals actually transmitted, both AIC and MDL determine that there are 9 transmitted signals. Therefore, although the recall for $k = 9$ is 1, the precision is very small. Furthermore, in almost all cases except for $k = 0$, both precision and recall are very small. Therefore, the results confirm that these mathematical schemes cannot be used for the asynchronous scenario, since they were proposed under the assumption that all transmitted signals are well synchronized.

In summary, we confirm that, for asynchronously transmitted signals, our ASensNet has remarkable detection performance with high reliability. By using deep-learning networks, our proposal outperforms the existing mathematical schemes.

6.3. Case with Different SNR for All Mobile Devices

In this section, our proposed scheme is evaluated under the scenario where the SNR values of transmitted signals are different. The results are obtained with various SNR ranges.

In Table 6, each element of the ‘SNRs [dB]’ column consists of the minimum SNR and the maximum SNR. Therefore, the element [6, 8] means that the SNR of a transmitted signal is within [6 dB, 8 dB].

Table 6. Precision and recall of our ASensNet with $d = 1$.

Precision		The Number of Transmitted Signals								
SNRs [dB]	0	1	2	3	4	5	6	7	8	9
[6, 8]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92
[6, 10]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
[6, 12]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92
[6, 14]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.91
[8, 10]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92
[8, 12]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
[8, 14]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
[8, 16]	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.91
[8, 18]	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.99	1.00	0.90
[8, 20]	1.00	1.00	0.98	1.00	0.99	0.99	0.99	0.98	1.00	0.91
Recall		The Number of Transmitted Signals								
SNRs [dB]	0	1	2	3	4	5	6	7	8	9
[6, 8]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[6, 10]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[6, 12]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[6, 14]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
[8, 10]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[8, 12]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[8, 14]	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
[8, 16]	1.00	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99
[8, 18]	1.00	0.89	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.98
[8, 20]	1.00	0.90	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.97

Table 6 shows the precision and recall with $d = 1$. (The precision and recall of AIC and MDL has a similar tendency to Tables 4 and 5, and thus these results are omitted in this paper.) The precision and recall present a similar tendency to Table 3. Hence, in almost all cases, our ASensNet guarantees that the precision and recall are larger than 0.9.

Therefore, this realistic evaluation confirms that our proposed scheme can accurately detect the number of asynchronously transmitted signals even if the SNRs of the signals vary.

Finally, Figure 8 shows case diagrams demonstrating how well each stream-sensing scheme follows the ground truth. Via the results, we also confirm that compared to other methods, our proposal can accurately estimate the number of survivors. Note that for Intel(R) Core(TM) i7-6900K CPU, the processing time for a case is 31.8 milliseconds, and the weight size of our ASensNet is 1.0 gigabyte. Therefore, our proposed scheme can operate in real time with an appropriate memory load.

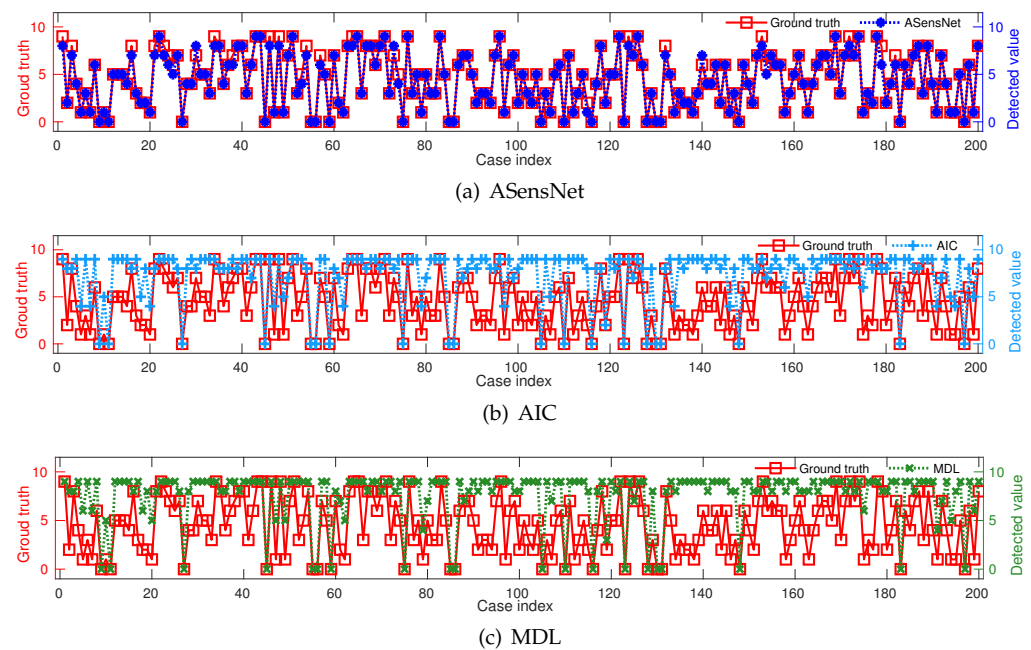


Figure 8. Case diagram like a real-time scenario.

7. Conclusions and Future Work

This work has proposed a machine-learning-based asynchronous stream-sensing scheme to estimate the number of asynchronously transmitted signals. For the scheme, we propose an asynchronous stream-sensing network (ASensNet) consisting of the AutoEncoder Extractor, the inception layer-based extractor, two linear extractors, and the classifier.

Through extensive simulation results, the large precision and recall confirm that our proposed ASensNet has notable detection performance with high reliability. Furthermore, the processing time is also fast even on a CPU (31.8 milliseconds for an Intel(R) Core(TM) i7-6900K CPU). Thanks to the fast processing speed and large precision and recall, while moving a disaster area, rescue UAVs with our ASensNet will be able to estimate the number of survivors per zone in real time.

This paper has demonstrated that it is possible to accurately detect the number of asynchronously transmitted signals, but the current ASensNet actually has a limitation in detecting a limited number of survivors: nine people. In addition, if the current system can be improved to detect the locations of devices, it will be more helpful to rescuers. Hence, in the future, we will provide various ASensNet variants for different numbers of antennas. In addition, based on our ASensNet, we will develop a high-quality survivor discovery protocol with which a rescue UAV not only detects the number of survivors around itself, but also estimates where the survivors are. Moreover, we will develop a high-quality survivor detection system with which a rescue UAV estimates the locations of devices around it by taking into account how the UAV will move through a disaster area to localize devices. Additionally, it can better respond to a situation where the SNR between a device and an UAV changes dynamically. Then, since deep learning could show a better performance that cannot be achieved using mathematical frameworks with assumptions, we will also study a deep-learning-based method that overcomes the theoretical limit and can detect more than $(N_R - 1)$ signals even with N_R antennas.

Author Contributions: Conceptualization, Y.K. and H.L.; methodology, Y.K.; software, Y.K. and H.L.; writing—original draft preparation, Y.K.; writing—review and editing, H.L.; supervision, H.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1G1A1007058).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. EMDAT. OFDA/CRED International Disaster Database, Université Catholique de Louvain—Brussels—Belgium: Number of Reported Disasters by Type. 2020. Available online: <https://ourworldindata.org/grapher/natural-disasters-by-type> (accessed on 1 December 2021).
2. Pregler, A. When COWs Fly: AT&T Sending LTE Signals from Drones. 2017. Available online: https://about.att.com/innovationblog/cows_fly (accessed on 1 December 2021).
3. Lavars, N. Verizon Trials Drones as Flying Cell Towers to Plug Holes in Internet Coverage. 2016. Available online: <https://newatlas.com/verizon-drones-internet-trials/45818/> (accessed on 1 December 2021).
4. Fortune Business Insights. *Small Drones Market Size, Share & Industry Analysis, By Type (Fixed-Wing, Rotary-Wing, and Hybrid/Transitional), By Power Source (Lithium-ion cells, Hybrid Fuel cells, and Solar cells), By Size (Micro, and Mini & Nano), By Application (Civil & Commercial, Military, Homeland Security, and Consumer) and Regional Forecast, 2015–2026*; Fortune Business Insights: Pune, India, 2020.
5. LeCun, Y.; Bengio, Y.; Hinton, G.E. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
6. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [[CrossRef](#)] [[PubMed](#)]
7. Goodfellow, I.J.; Bengio, Y.; Courville, A.C. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
8. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [[CrossRef](#)]
9. Purwins, H.; Li, B.; Virtanen, T.; Schlüter, J.; Chang, S.; Sainath, T. Deep Learning for Audio Signal Processing. *IEEE J. Sel. Top. Signal Process.* **2019**, *13*, 206–219. [[CrossRef](#)]
10. O’Shea, T.; Hoydis, J. An Introduction to Deep Learning for the Physical Layer. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 563–575. [[CrossRef](#)]
11. Mao, Q.; Hu, F.; Hao, Q. Deep Learning for Intelligent Wireless Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2595–2621. [[CrossRef](#)]
12. Liao, J.; Zhao, J.; Gao, F.; Li, G.Y. A Model-Driven Deep Learning Method for Massive MIMO Detection. *IEEE Commun. Lett.* **2020**, *24*, 1724–1728. [[CrossRef](#)]
13. Huang, H.; Yang, J.; Huang, H.; Song, Y.; Gui, G. Deep Learning for Super-Resolution Channel Estimation and DOA Estimation Based Massive MIMO System. *IEEE Trans. Veh. Technol.* **2018**, *67*, 8549–8560. [[CrossRef](#)]
14. Nachmani, E.; Marciano, E.; Lugosch, L.; Gross, W.J.; Burshtein, D.; Be’ery, Y. Deep Learning Methods for Improved Decoding of Linear Codes. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 119–131. [[CrossRef](#)]
15. Vyas, M.R.; Patel, D.K.; Lopez-Benitez, M. Artificial neural network based hybrid spectrum sensing scheme for cognitive radio. In Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Montreal, QC, Canada, 8–13 October 2017; pp. 1–7. [[CrossRef](#)]
16. Han, D.; Sobabe, G.C.; Zhang, C.; Bai, X.; Wang, Z.; Liu, S.; Guo, B. Spectrum sensing for cognitive radio based on convolution neural network. In Proceedings of the 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 14–16 October 2017; pp. 1–6. [[CrossRef](#)]
17. Liu, C.; Wang, J.; Liu, X.; Liang, Y. Deep CM-CNN for Spectrum Sensing in Cognitive Radio. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2306–2321. [[CrossRef](#)]
18. Cheng, Q.; Shi, Z.; Nguyen, D.N.; Dutkiewicz, E. Sensing OFDM Signal: A Deep Learning Approach. *IEEE Trans. Commun.* **2019**, *67*, 7785–7798. [[CrossRef](#)]
19. Paisana, F.; Selim, A.; Kist, M.; Alvarez, P.; Tallon, J.; Bluemm, C.; Puschmann, A.; DaSilva, L. Context-aware cognitive radio using deep learning. In Proceedings of the 2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), Baltimore, MD, USA, 6–9 March 2017; pp. 1–2. [[CrossRef](#)]
20. Xie, J.; Fang, J.; Liu, C.; Yang, L. Unsupervised Deep Spectrum Sensing: A Variational Auto-Encoder Based Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5307–5319. [[CrossRef](#)]
21. Wax, M.; Kailath, T. Detection of signals by information theoretic criteria. *IEEE Trans. Acoust. Speech Signal Process.* **1985**, *33*, 387–392. [[CrossRef](#)]
22. Fishler, E.; Grossmann, M.; Messer, H. Detection of signals by information theoretic criteria: general asymptotic performance analysis. *IEEE Trans. Signal Process.* **2002**, *50*, 1027–1036. [[CrossRef](#)]
23. Akaike, H. Information theory and an extension of the maximum likelihood principle. In Proceedings of the 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, 2–8 September 1971.

24. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Autom. Control* **1974**, *19*, 716–723. [CrossRef]
25. Schwarz, G. Estimating the dimension of a model. *Ann. Stat.* **1978**, *6*, 461–464. [CrossRef]
26. Rissanen, J. Modeling by shortest data description. *Automatica* **1978**, *14*, 465–471. [CrossRef]
27. 3GPP. Overview of 3GPP Release 13. 2014. Available online: https://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/ (accessed on 1 December 2021).
28. Fu, R.; Al-Absi, M.A.; Kim, K.; Lee, Y.S.; Al-Absi, A.A.; Lee, H. Deep Learning-Based Drone Classification Using Radar Cross Section Signatures at mmWave Frequencies. *IEEE Access* **2021**, *9*, 161431–161444. [CrossRef]
29. Jamil, S.; Abbas, M.S.; Roy, A.M. Distinguishing Malicious Drones Using Vision Transformer. *AI* **2022**, *3*, 260–273. [CrossRef]
30. 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA)*; Physical Layer Procedures (Release 14); 3GPP Organizational Partners, Sophia Antipolis CEDEX: Valbonne, France, 2018.
31. IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*; IEEE: New York, NY, USA, 2012.
32. Telesystem Innovations Inc. *LTE in a Nutshell: The Physical Layer*; Telesystem Innovations Inc.: Markham, ON, Canada, 2010.
33. NTT DOCOMO. Uplink multiple access schemes for NR. In Proceedings of the R1-165174, 3GPP TSG-RAN WG1 Meeting 85, Nanjing, China, 23–27 May 2016.
34. IEEE 802.11ac-2013. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*; IEEE: New York, USA, 2013.
35. ITU. Guidelines for Evaluation of Radio Interface Technologies for IMT-Advanced. 2009. Available online: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2135-1-2009-PDF-E.pdf (accessed on 1 December 2021).
36. ETSI. *Broadband Radio Access Networks (BRAN); 5 GHz High Performance RLAN*; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive; European Telecommunications Standards Institute, Sophia Antipolis CEDEX: Valbonne, France, 2015.
37. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.
38. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *arXiv* **2015**, arXiv:1512.00567.
39. Tran, D.; Bourdev, L.D.; Fergus, R.; Torresani, L.; Paluri, M. Learning Spatiotemporal Features with 3D Convolutional Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Piscataway, NJ, USA, 2015; pp. 4489–4497. [CrossRef]
40. Deng, L. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Trans. Signal Inf. Process.* **2014**, *3*, E2. [CrossRef]
41. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
42. CS231n. Multiclass Support Vector Machine Loss. Available online: <https://cs231n.github.io/linear-classify/> (accessed on 1 December 2021).
43. MathWorks. Calling MATLAB from Python. Available online: <https://www.mathworks.com/help/matlab/matlab-engine-for-python.html> (accessed on 1 December 2021).
44. Jiang, P.; Wang, T.; Han, B.; Gao, X.; Zhang, J.; Wen, C.K.; Jin, S.; Li, G.Y. AI-aided Online Adaptive OFDM Receiver: Design and Experimental Results. *IEEE Trans. Wireless Commun.* **2021**, *20*, 7655–7668. [CrossRef]
45. Jiang, P.; Wen, C.; Jin, S.; Li, G.Y. Dual CNN-Based Channel Estimation for MIMO-OFDM Systems. *IEEE Trans. Commun.* **2021**, *69*, 5859–5872. [CrossRef]
46. Jankowski, M.; Gündüz, D.; Mikolajczyk, K. Wireless Image Retrieval at the Edge. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 89–100. [CrossRef]
47. Kurka, D.B.; Gündüz, D. DeepJSCC-f: Deep Joint Source-Channel Coding of Images with Feedback. *IEEE J. Sel. Areas Inf. Theory* **2020**, *1*, 178–193. [CrossRef]
48. Bourtsoulatte, E.; Kurka, D.B.; Gündüz, D. Deep Joint Source-Channel Coding for Wireless Image Transmission. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 567–579. [CrossRef]
49. Jang, J.; Yang, H.J. Deep Reinforcement Learning-Based Resource Allocation and Power Control in Small Cells With Limited Information Exchange. *IEEE Trans. Veh. Technol.* **2020**, *69*, 13,768–13,783. [CrossRef]
50. Xie, H.; Qin, Z.; Li, G.Y.; Juang, B. Deep Learning Enabled Semantic Communication Systems. *IEEE Trans. Signal Process.* **2021**, *69*, 2663–2675. [CrossRef]
51. Weng, Z.; Qin, Z. Semantic Communication Systems for Speech Transmission. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2434–2444. [CrossRef]
52. Maximov, M.; Elezi, I.; Leal-Taixé, L. CIAGAN: Conditional Identity Anonymization Generative Adversarial Networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5446–5455.
53. Movshovitz-Attias, Y.; Toshev, A.; Leung, T.K.; Ioffe, S.; Singh, S. No Fuss Distance Metric Learning Using Proxies. In Proceedings of the IEEE International Conference on Computer Vision ICCV, Venice, Italy, 22–29 October 2017; pp. 360–368.
54. Lee, H.; Kim, M.U.; Kim, Y.; Lyu, H.; Yang, H.J. Development of a Privacy-Preserving UAV System With Deep Learning-Based Face Anonymization. *IEEE Access* **2021**, *9*, 132652–132662. [CrossRef]