Adaptive Uplink Data Compression in Spectrum Crowdsensing Systems

Yijing Zeng, *Student Member, IEEE*, Roberto Calvo-Palomino, *Member, IEEE*, Domenico Giustiniano, *Senior Member, IEEE*, Gerome Bovet, and Suman Banerjee, *Fellow, IEEE, ACM*

***Abstract*— Understanding spectrum activity is challenging when attempted at scale. The wireless community has recently risen to this challenge in designing spectrum monitoring systems that utilize many low-cost spectrum sensors to gather large volumes of sampled data across space, time, and frequencies. These crowdsensing systems are limited by the uplink bandwidth available to backhaul the raw in-phase and quadrature (IQ) samples and power spectrum density (PSD) data needed to run various applications. This paper presents FlexSpec, a framework based on the Walsh-Hadamard transform to compress spectrum data collected from distributed and low-cost sensors for real-time applications. This transformation allows sensors to significantly save uplink bandwidth thanks to its inherent properties both when it is applied to IQ and PSD data. Additionally, by leveraging a feedback loop between the sensor and the edge device it connects to, FlexSpec carefully adapts the compression ratio over time to changes in the spectrum and different applications, jointly considering data size, application performance, and spectrum variations. We experimentally evaluate FlexSpec in several applications. Our results show that FlexSpec is particularly suitable for IoT transmissions and signals close to the noise floor. Compared with prior work, FlexSpec provides up to 7× more reduction of uplink data size for signal detection based on PSD data, and reduces up to 6× to 8× the number of undecodable messages for IQ sample decoding.**

***Index Terms*— Spectrum crowdsensing, adaptive data compression, fast Walsh-Hadamard transform, feedback loop.**

Manuscript received 18 December 2021; revised 14 August 2022 and 30 October 2022; accepted 18 January 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Chen. This work was supported in part by the Fundación IMDEA Networks (IMDEA) through the Madrid Regional Government Técnicas Avanzadas para Potenciar la Inteligencia de las Redes "(TAPIR-CM) Project" under Grant S2018/TCS-4496; in part by the Ministry of Economic Affairs and Digital Transformation through the Project Machine Learning-based Privacy Preserving Analytics for 6G Mobile Networks (MAP-6G) under Grant TSI-063000-2021-63. The work of Yijing Zeng and Suman Banerjee was supported in part by the U.S. National Science Foundation under Grant CNS-2112562, Grant CNS-2107060, Grant CNS-2003129, Grant CNS-1838733, and Grant CNS-1647152; and in part by the U.S. Department of Commerce under Award 70NANB21H043. A short version of this paper was presented in the IEEE DySPAN 2021 [DOI: 10.1109/DySPAN53946.2021.9677075]. *(Corresponding author: Yijing Zeng.)*

Yijing Zeng and Suman Banerjee are with the Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: yijingzeng@cs.wisc.edu; suman@cs.wisc.edu).

Roberto Calvo-Palomino is with the Department of Signal Theory and Communications and Telematic Systems and Computing, Universidad Rey Juan Carlos, 28933 Madrid, Spain (e-mail: roberto.calvo@urjc.es).

Domenico Giustiniano is with the IMDEA Networks Institute, 28918 Leganés, Spain (e-mail: domenico.giustiniano@imdea.org).

Gerome Bovet is with Armasuisse, 3003 Bern, Switzerland (e-mail: Gerome.Bovet@armasuisse.ch).

Digital Object Identifier 10.1109/TNET.2023.3239378

I. INTRODUCTION

MEASURING and understanding spectrum activity at scale — time, space, and frequency — is challenging. Traditionally, a few high-end spectrum sensors have been used to understand such activity. In recent years, the community has devoted effort to building low-cost spectrum sensors that many users can potentially deploy at a much grander scale, e.g. [24], [30], [50]. These spectrum crowdsensing systems have an architecture where spectrum data can be continuously obtained from a large number of very low-cost spectrum sensors (e.g., the $\sim$ \$25 RTL-SDR) and aggregated in cloud-hosted backends for further analytics by different applications. A high-level illustration of the architecture is presented in Fig. 1, with low-cost sensors that transmit their data to edge devices and several applications that the end-user can select. In particular, the low-cost nature of the sensor and its easy integration into an existing cloud-hosted backend often motivates more participants to join this effort for the greater good of the community.

The two common types of spectrum data of interest to various use cases are power spectral density (PSD) data and in-phase and quadrature (IQ) data. The former is helpful to various applications in understanding spectrum utilization and occupancy, while the latter helps analyze wireless protocol behaviors and decode messages. For each data form, the sensor can quickly generate a large volume of data, which eventually needs to be backhauled and stored remotely. For instance, the RTL-SDR generates 2.4M IQ samples per second, which results in a data rate of 153.6 Mbps per sensor, often beyond the capacity of any common backhaul network.

Prior efforts have proposed different spectrum data compression methods for streaming spectrum data. Electrosense leverages lossless compression to pack data more compactly [29], [30]. Unfortunately, this approach cannot compress much due to the nature of spectrum data. Airpress [51] and SparSDR [22] propose lossy compression methods for PSD and IQ data, respectively. They achieve greater compression ratios than lossless compression. However, SparSDR depends on setting a specific threshold to determine whether there is any wireless activity and whether such samples should be uploaded. In the meantime, Airpress uses a fixed compression ratio that does not allow the application to adapt to changes in the received spectrum data. As we will show in § II, these cause missing important information in the compressed spectrum data and lack of adaptation with respect to the application needs, which is especially relevant both for Internet of Things (IoT) wireless transmissions and signals close to the noise floor.

Adaptive Compression of Spectrum Data: We propose a framework, FlexSpec, which performs adaptive compression of spectrum data uploaded from distributed spectrum sensors in such crowdsensing systems. FlexSpec addresses the limitations of prior work and is based on two components, a lightweight real-time lossy compression method and an application-oriented compression adaption scheme. Our real-time lossy compression runs in the sensor with feedback received from the edge device.

FlexSpec's compression method is based on fast Walsh-Hadamard transformation (FWHT). FWHT has been applied in a variety of other fields. It received large attention in the past because its low-computational cost and simplicity of transform bases. For spectrum data compression, our key idea is that we *apply this transformation on single spectrum data record* (either PSD or IQ) and only *retain significant readings in the transferred domain*. To our knowledge, we are the first to apply FWHT on spectrum data compression and show why it works well on both PSD and IQ data. The core reason is that the transferred domain of either PSD or IQ data captures the features of the data. The bases of FWHT capture the bandwidth-limited feature of signals in PSD data, and the transferred domain for the IQ data is a variant of the frequency domain. In addition, this method has a highly adjustable compression ratio, meaning the more appropriate choice is available when adapting the compression ratio.

In FlexSpec, the amount of compression to be applied depends specifically on the application of interest. Consider an application like spectrum utilization measurements over a wide area that operates over PSD data. The samples in this setting can be compressed in a lossy manner quite substantially because the results can gracefully degrade with loss in fidelity of the data. If the uplink capacity is low and the energy measurements have reduced fidelity, the application can still provide reasonable estimates from them. However, if we consider another application — decoding broadcast ADS-B messages from aircraft transponders that allow them to be tracked — it might not be equally amenable to reduce fidelity of information. Therefore, aggressively lossy compression might not be suitable in this case. Consequently, FlexSpec aims to support dynamic selection of the compression ratio to support a variety of spectrum applications according to the user's interests and is suited to work with live streaming of spectrum data (consumed by the running application).

FlexSpec introduces an application-oriented compression ratio adaptation scheme to enable dynamic compression ratio adaptation for different applications and changes in the captured spectrum. The essence of this scheme is that, given the desired application performance metric(s), we *introduce a feedback loop between the edge device and the sensor to jointly decide the compression ratio based on the target application's performance on current compressed spectrum data.* As a result, FlexSpec can consistently optimize the desired application performance but send as a little amount of data as possible in a varying spectrum band, which is impossible for prior work.

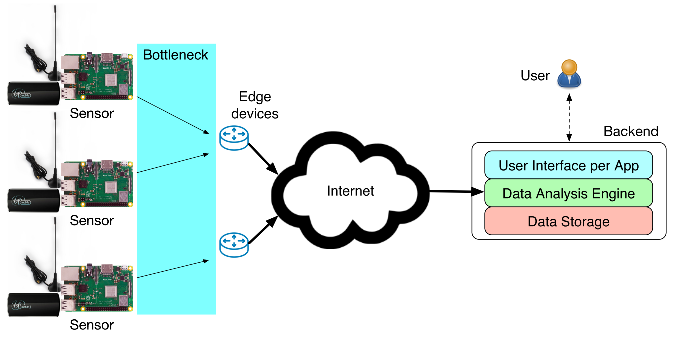


Fig. 1. Illustration of spectrum crowdsensing system supporting a variety of applications; the bottleneck is the backhaul connection from the sensors.

Our key contributions are the following:

- We propose FlexSpec, *a framework for adaptive uplink data compression in spectrum crowdsensing systems*. It supports live streaming of spectrum data for various applications, e.g., signal detection, IQ decoding.
- FlexSpec's compression algorithm *works well for various applications on both PSD and IQ data*. For spectrum visualization of PSD data, our method introduces up to 5 dB less reconstruction error than prior work. We have up to a 20% better false negative rate for detecting narrowband signals with PSD data when the compression ratio is high. With IQ data, we can decode ADS-B signals received from aircrafts with up to 70% higher successful decoding rate when the compression ratio is medium. We can perform signal technology classification distinguishing between WiFi and LTE with up to 15% higher accuracy than prior work.
- FlexSpec's application-oriented compression ratio adaptation scheme *dynamically adjusts the compression ratio based on the application's performance and current spectrum usage*. As an example, for signal detection based on PSD data, FlexSpec misses less than 1 FM channel (out of 9 total) in the FM radio band when there is spectrum activity. In the meantime, it has a $7\times$ more reduction in uplink data size when there is no spectrum activity compared with prior work. For IQ decoding, FlexSpec reduces $6\times$ to $8\times$ the number of undecodable messages compared with prior work with a similar compression ratio.

II. Challenges

We aim to compress spectrum data, containing either PSD or IQ readings, without affecting analytics performed in the backend. We consider either the PSD or IQ readings in a single data record, i.e., a row vector $\mathbf{x} \in \mathbb{R}^N$ for PSD or $\mathbf{x} \in \mathbb{C}^N$ for IQ, and ignore meta-data like GPS information because the meta-data is significantly smaller than PSD or IQ readings in size. Because the spectrum data is noisy, saving large data volume is only possible with lossy compression.

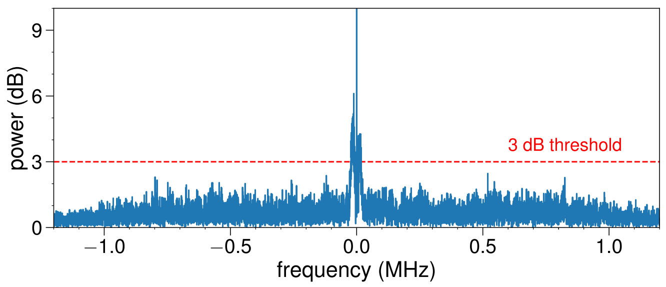


Fig. 2. PSD of a sample ADS-B signal. The bandwidth of the signal determined by the 3 dB threshold above the noise level is 80 KHz, but important information is lost with this threshold.

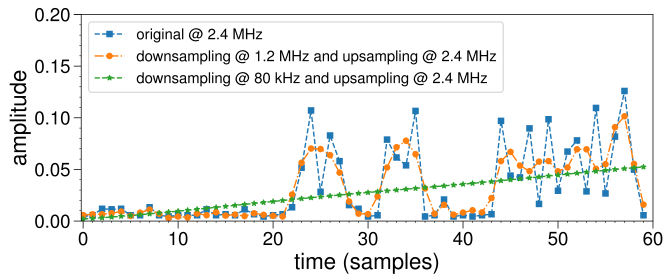


Fig. 3. Amplitude of ADS-B signal over time. Downsampling causes detail information loss, which makes ADS-B signal hard to be decoded.

However, doing it on low-cost sensors, e.g., RTL-SDR [7] with Raspberry Pi 4 (RBPi-4) [5], raises the following challenges to the spectrum data compression techniques.

Limited computing resources. Low-cost sensors usually have a relatively small amount of memory (8 GB for RBPi-4) and only a few CPU cores (4 cores for RBPi-4). Therefore, unlike prior work with high computational cost ([32], [48]), the compression technique in our case should have low time/space complexity such that it can compress streaming spectrum data in the resource-limited low-cost sensors in real-time.

Spectrum occupancy and wideband signals. Wireless signals have bandwidth B that can vary largely. In contrast, low cost-sensors have limited support in terms of sampling rate S. This has implications on the compression ratio strategy. For narrowband signals ($S \geq B$), methods that estimate *spectrum occupancy* using a power threshold in the frequency domain have been used to select the active frequency bins to be backhauled, as done in prior work by SparSDR [22]. While this idea is simple, measuring the spectrum occupancy is often ineffective. For instance, the International Telecommunication Union (ITU) report on spectrum occupancy suggests declaring the channel as busy if it is 3-5 dB above the noise level [38]. Fig. 2 shows the PSD of a ADS-B signal trace sent by an aircraft and captured by RTL-SDR with 2.4 MHz sampling rate. Following the above ITU report, its spectrum occupancy would be 80 KHz. However, the signal cannot be decoded at all with this 80 KHz band. The reason is that the modulation used in ADS-B causes leakages outside the 80 kHz band and near the noise floor, yet fundamental for message decoding. Besides, even in those cases when spectrum occupancy measurement in SparSDR is effective on narrowband signals, the output sequence could be further compressed depending on the application's requirements, a chance ignored by past work. On the other hand, the captured signals can be wideband ($S < B$) for numerous cases, as S is relatively small in low-cost sensors (e.g. $S = 2.4$ MHz in RTL-SDR). In this case, spectrum occupancy is ineffective, as all frequency bins must be backhauled. In summary, a different approach is needed that does not depend on spectrum occupancy measurements.

Detail information is important for IQ data. Airpress compresses the PSD data with the goal of spectrum summarization [51]. It applies the Haar wavelet transform to get the approximation and detail coefficients, and assumes that detail coefficients are not crucial for spectrum summarization. Although this assumption is valid for PSD data, it becomes invalid for IQ data. For example, if we would like to decode IQ samples of FM radio signal with a reasonable SNR, the minimal possible compression ratio of 2 will directly lead to noisy audio that cannot be perceived by human. Therefore, for IQ data, we need a method that balances approximation and detail during the compression. Note that a byproduct of completely omitting detail coefficients in Airpress is that it can only support exponential compression ratios in the form of 2^i $(i = 0, \ldots, n)$, given a PSD data with $N = 2^n$ readings. The limited choices in compression ratios makes fine-grained compression ratio adaptation impractical. By balancing approximation and detail, we can potentially support linear compression ratios in the form of N/i $(i = 1, \ldots, N)$ for fine-grained compression ratio adaptation. In addition, downsampling, a naive approach to compress IQ data, also loses the detail information. For instance, Fig. 3 shows the amplitude of reconstructed ADS-B signal after downsampling/upsampling, and compares with the original signal. After downsampling to 1.2 MHz, we can see that the preambles needed to decode ADS-B signal, which last from time 20 to 40, are already distorted. The four spikes becomes two. If the signal is downsampled to 80 kHz as the spectrum occupancy result suggested, not only are the preambles distorted, but the signal duration that carries messages become meaningless.

Difficulty in deciding the compression ratio. Lossless compression does not save much data volume due to its inability to capture the properties of the noisy spectrum data. Our experience with Gzip (used in Electrosense [30]) shows that it can only reduce data size by 10% for IQ data. Lossy compression can instead allow for a balance of the information loss and compression ratio. However, how much information loss is tolerable highly depends on the specific application that is streamed at the current time. For example, if the application is to know whether a channel is occupied based on PSD data, it tolerates relatively high information loss. If the application is to decode messages from IQ samples, we may only tolerate little information loss. In addition, significant utilization change of the spectrum band, e.g., from idle to active, and discrepancy of activities in different bands may also result in the difference of tolerable information loss. Prior work did not study this problem.

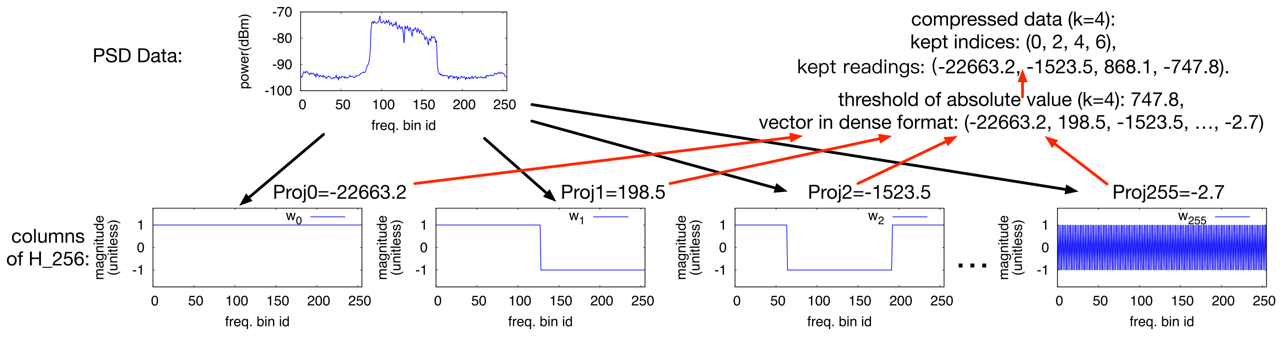

PSD Data:
power(dBm)
freq. bin id
compressed data (k=4):
kept indices: (0, 2, 4, 6),
kept readings: (-22663.2, -1523.5, 868.1, -747.8).
threshold of absolute value (k=4): 747.8,
vector in dense format: (-22663.2, 198.5, -1523.5, ..., -2.7)
Proj0=-22663.2
Proj1=198.5
Proj2=-1523.5
Proj255=-2.7
columns of H_256:
magnitude (unitless)
w0
w1
w2
w255

Fig. 4. Illustration of FlexSpec's compression algorithm for PSD data. We apply FWHT on single PSD data records and only keeps significant readings in the transferred domain, which are encoded as a sparse vector.

III. Compression Algorithm

In this section, we present FlexSpec's spectrum data compression algorithm. Our method applies a low-cost transformation, fast Walsh-Hadamard transform (FWHT), on single spectrum measurement and only keeps significant readings in the transferred domain, where both the features of PSD and IQ data can be captured. We first review the principles of the Walsh-Hadamard transform and then present our compression method and explain the ideas behind its efficacy for spectrum data compression.

A. Primer on Walsh-Hadamard Transform

The Walsh-Hadamard transform [35] is a generalized form of the Fourier transform. Unlike the Fourier transform, which uses complex sine and cosine functions as the orthogonal basis, the Walsh-Hadamard transform uses real Walsh functions as the orthogonal basis, containing only -1 and 1. Formally, let H_{2^n} denote the Hadamard matrix with row/column size equals 2^n and $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. We have the following relationship between $H_{2^{n+1}}$ and H_{2^n}:

$$H_{2^{n+1}} = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix}, \quad n = 1, 2, 3, \dots. \tag{1}$$

Therefore, each row/column of H_{2^n} is a Walsh function, which contains only -1 and 1, and is orthogonal to each other. Let the data with $N = 2^n$ readings as a row vector $\mathbf{x}$. Its Walsh-Hadamard transform is defined as

$$\mathbf{X} = \mathbf{x} \cdot H_{2^n}. \tag{2}$$

The corresponding inverse transform is computed as

$$\mathbf{x} = \frac{1}{N}\mathbf{X} \cdot H_{2^n}. \tag{3}$$

Similar to the Fourier transform, which has a fast butterfly algorithm with $O(N\log N)$ time complexity, i.e., fast Fourier transform (FFT), the Walsh-Hadamard transform also has a fast butterfly algorithm with $O(N\log N)$ time complexity, i.e., FWHT. The difference is that FFT involves $N\log_2 N$ complex add operations and $\frac{1}{2}N\log_2 N$ complex multiplication operations, but FWHT only involves $N\log_2 N$ real add operations.

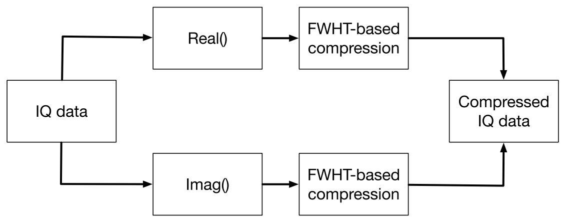


Fig. 5. Illustration of FlexSpec's compression algorithm for IQ data. FWHT based compression is applied to in-phase and quadrature readings separately.

B. PSD Data Compression

We compress PSD data $\mathbf{x} \in \mathbb{R}^N$ and the number of kept readings k as follows. We ignore the readings in the transferred domain close to 0 and encode this dense vector in the transferred domain as a sparse vector.[1] More specifically, we compute the FWHT of the PSD data and then record the absolute values in the transferred domain. Next, we get the kth largest absolute value of readings in the transferred domain using Quick Select [20] algorithm, where k is selected by the application. Last, we get all the kept readings in the transferred domain and encode them as a sparse vector. To decompress, we first convert the compressed data into the dense format and then compute the inverse transform with Eq. (3).

Note that we can use a bit mask to physically encode (compress) the indices of kept readings, which saves even more space when the compression ratio is small. For example, for a PSD measurement with 256 frequency bins, its indices can be encoded into a 256-bit field. If the reading of an index is kept, then we can set the corresponding bit in the field as 1. Otherwise, we keep the corresponding bit in the field as 0. Nevertheless, it is a minor optimization and only beneficial when the compression ratio is smaller than 16, i.e., the size of kept indices is larger than the 256-bit field.

C. IQ Data Compression

We now extend our compression algorithm on PSD data to IQ data. Fig. 5 illustrates the process. We first get the real part and the imaginary part of IQ data, and then apply the

[1] A dense vector is an array that contains mostly non-zero values, and a sparse vector is an array that contains mostly zeros. For the representation, a dense vector presents all values of the vector, while a sparse vector is described as two parallel arrays, indices and values.

FWHT-based compression algorithm to the in-phase (I) reading sequence and the quadrature (Q) reading sequence separately.

Other options are possible, too, e.g., directly use complex FWHT, convert I and Q coordinates to amplitude and phase polar coordinates first then apply the algorithm to amplitude reading sequence and phase reading sequence separately. The best option in terms of compression ratio varies for different modulation scheme of the signal (e.g. if the signal is purely phase modulated, convert I and Q coordinates to amplitude and phase polar coordinates is very desirable). However, our method (1) has lower time complexity so that it can work on low-cost sensors in real-time because it involves real number addition and subtraction only, and (2) the transferred domain is a variant of the frequency domain, called sequency domain [19], so it captures the features of IQ data in this analogous frequency domain.

D. Suitability for Spectrum Compression

Although FWHT is commonly applied in fields like image compression [42] and video compression [23], our main contribution is to show its application on spectrum data for the first time. At a high level, FWHT can capture the features of both PSD and IQ data. In fact, the bases of the transferred domain for PSD data represent one or more (frequency domain) bandwidth-limited signals (c.f. Fig. 4), and the transferred domain of IQ data is a variant of the frequency domain.

PSD. Fig. 4 illustrates how FlexSpec's compression algorithm captures the features of a single PSD measurement with 256 frequency bins. In Fig. 4, we reorder the columns of H_{256} so that each column/Walsh function, denoted as $\mathbf{w}_i, 0 \leq i \leq 255$, has i sign changes. We notice that $\mathbf{w}_0$ and the corresponding Projection0 represent the average power of the PSD measurement, which highly depends on the noise floor value. Starting from $\mathbf{w}_1$, each Walsh function equals 1 for half of the frequency bins and equals -1 for the other half. Therefore, if the projection on $\mathbf{w}_i$ is bigger than 0, it means there are probably notable signal(s) in the frequency bins whose corresponding values are 1. Otherwise, there are probably notable signal(s) in the frequency bins whose corresponding values are -1. For example, if Projection1 is bigger than 0, there are probably notable signal(s) from frequency bin 0 to 127. If Projection2 is smaller than 0, there are probably notable signal(s) from frequency bin 64 to 191. In both cases, the bases represent one or more bandwidth-limited signals regardless of whether the projection is positive. Moreover, we keep the projections that have the largest absolute values rather than keep the projections whose corresponding Walsh functions have a fewer number of sign changes. As a result, our compression algorithm can preserve both relatively wideband signals, e.g., ones captured by $\mathbf{w}_1$ and $\mathbf{w}_2$, and narrowband signals, e.g., ones captured by $\mathbf{w}_{255}$.

Compared with Airpress [51], our compression algorithm supports more possible compression ratios (any value in the form of $N/i, i = 1, \ldots, N$), which is better suited for adaptive compression scenarios. Our FWHT-based compression on PSD data also brings two additional advantages over Airpress. First, in Airpress, all approximation coefficients are retained, but all detail coefficients are ignored after the Haar wavelet transform. As a result, our method can capture more detailed information when the same compression ratio is used, i.e., less information loss. Second, our algorithm is more friendly for narrowband signals. The reason is that in Airpress, each PSD reading is approximated by the corresponding approximation coefficient, which is the average of consecutive 2^l (l is the level of transformation) PSD readings. Therefore, the bandwidth of narrowband signals can be significantly enlarged, but their power can be drastically reduced.

IQ. Besides capturing features in the sequency domain with a low complexity (§ III-C), we consider our compression algorithm as a good alternative for FFT-based compression due to the following reasons. First, as FWHT's orthogonal bases are square waves rather than sine waves, our algorithm will retain fewer projections for signals whose baseband waveform is more similar to square waves than FFT-based compression, e.g., SparSDR. Second, compared with complex number FFT, FWHT on I and Q readings respectively saves more than half the number of real-valued operations (including addition or multiplication). It means it has higher time efficiency and is a better fit for low-cost devices. Last, we offer an opportunity to compress I and Q readings separately. I and Q readings are usually equally crucial in traditional modulation schemes. However, recent advances in deep learning-based communication design show that in a learning-based modulation scheme, the constellation points can be asymmetric around the origin point [28], so the importance of I and Q readings can differ from each other. In addition, the relative importance of each basis within I and Q readings can also differ, as modulation schemes may exploit I and Q readings differently. In these cases, compressing I and Q readings separately is more beneficial than compressing them together.

E. Overhead and Complexity

The nominal compression ratio of the algorithm is N/k, but we cannot achieve this due to the overhead of indices of sparse representation. Moreover, because the average time complexity and the space complexity of Quick Select are $O(N)$ and $O(1)$, respectively, the time complexity and the space complexity are $O(N\log N)$ and $O(N)$.

IV. Compression Ratio Adaptation

FlexSpec is designed to efficiently compress data transmitted in real-time by the sensor and used by the application specified by the user. We define the user as any person with access to the applications implemented by the system. In our view, the user will have control of the sensor[2] and dynamically select a specific application to run. Although FWHT-based compression could also be applied to historical data in the backend,[3] for live consumption of data, additional

[2]Given the spectrum sensors' limited hardware and software capabilities, we do not consider the extension to multiple users controlling the same sensor realistic.

[3]If the same data was already compressed based on the application's needs, storing this data for historical processing could use the same or higher ratio than living data consumption.

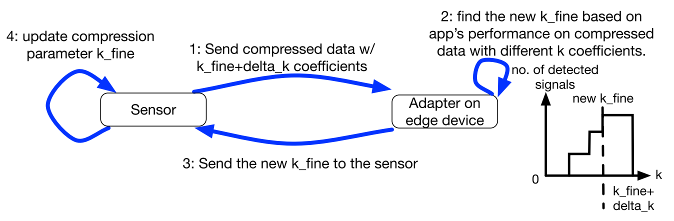

4: update compression parameter k_fine
1: Send compressed data w/ k_fine+delta_k coefficients
2: find the new k_fine based on app's performance on compressed data with different k coefficients.
Sensor
Adapter on edge device
3: Send the new k_fine to the sensor
no. of detected signals
new k_fine
0
k
k_fine+ delta_k

Fig. 6. Illustration of FlexSpec's application-oriented compression ratio adaptation scheme.

optimization in compression ratio is possible. We observe that *the tolerable information loss varies for different applications, and changes in spectrum utilization also lead to variations of tolerable information loss,* which is the basis of compression ratio adaptation. Previous works do not explicitly consider adaptive compression because they are designed for specific applications. In this section, we show how the compression ratio can be dynamically adapted in FlexSpec so that the information loss can be adjusted to the variations of spectrum utilization and satisfy the requirement of different applications.

Overall, FlexSpec's application-oriented compression ratio adaptation scheme works as follows. *We introduce a feedback loop between an adapter and the sensor to jointly decide the compression coefficient k based on the application's performance on current compressed data.* The adapter decides and sends the appropriate compression coefficient k_{fine} to be used by the sensor given the compressed data (we will explain how to decide it shortly). Then, to be aware of changes in the spectrum utilization, the sensor adds some redundancy to the uplink data, i.e., it uploads the compressed data with $k_{fine} + \Delta k$ coefficients. The amount of redundancy added is rich enough to affect the application's performance when the utilization of the spectrum is changed. Given the compressed data with $k_{fine} + \Delta k$ coefficients, the adapter decides the new k_{fine} based on the application's performance. It outputs the smallest k, such that the difference between the application's performance on the compressed data with k coefficients and that on the compressed data with $k_{fine} + \Delta k$ coefficients is tolerable in terms of some performance metrics specified by the user. For example, the number of detected signals for signal detection application. Fig. 6 illustrates this scheme. Note that naively applying a feedback loop does not work. The reason is that the application's performance on the compressed data with a larger compression coefficient k cannot be inferred based on the current compressed data, which means the feedback loop cannot decrease the compression ratio. As a result, adding a small redundancy in our proposed feedback loop is critical.

Initialization. During the initialization phase, if the uplink cannot handle uncompressed data, the sensor can be configured to compute the initial compression coefficient to be used, k_{coarse}, by itself. For PSD data, we compute k_{coarse} in order to have at most a 5 dB error for every frequency bin between the reconstructed data and the uncompressed data. For IQ data, we start with a compression ratio of 2 directly.

Reduce computation cost at the adapter. When the number of sensors that an adapter controls is large, there is a need to further reduce the computation cost at the adapter. A simplified version of the adapter's logic is to only evaluate three options for the new compression ratio, which are $k_{fine} - \Delta k, k_{fine}$, and $k_{fine} + \Delta k$. This simplified logic can be parallelized easily.

Reduce k_{fine} fluctuations. The adapter's logic computes k_{fine} on every compressed data sent from the sensor. However, the result of k_{fine} may fluctuate (usually towards a more aggressive compression coefficient) due to some random noise in the data. Therefore, we reduce the fluctuations (as well as the downlink traffic) and output the new k_{fine} on every m compressed data records (usually around 10). The adapter selects the largest one (most conservative one) from the m computed new compression coefficients because the logic at the adapter is relatively aggressive (see the limitation below).

Deciding Δk. A large Δk needs a higher uplink data rate, but it can handle bursty changes in spectrum utilization. One can determine Δk by replaying a synthetic data trace, which adds more significant variations to a real-world (uncompressed) data trace, to the system before supporting live streaming data, and choose from $1/2^i$ of the segment size. In practice, we find that Δk within the range of 1/8 to 1/16 of the segment size works well in balancing the two sides for the applications we tested.

Multiple application performance metrics. It is straightforward to incorporate multiple application performance metrics. For example, we can easily add the center frequency, the bandwidth of detected signal(s), etc., into the toy example of signal detection application in Fig. 6 by outputting the highest k_{fine} for different metrics. The computations of k_{fine} for different metrics are also parallelized.

Limitation. A limitation of this scheme is that there is no theoretical guarantee on the gap between the application's performance on compressed data and uncompressed data. The adapter can output a very aggressive compression coefficient such that the redundancy added is insufficient. Further, uploading uncompressed data, even only for a small portion of all the traffic, is undesirable or, depending on the available bandwidth, even impossible. As a result, we cannot access to the ground truth of the application's performance to calibrate our system. If this assumption is relaxed, we can upload a small portion of data uncompressed to calibrate the application performance. However, we empirically find that the application's performance always converges to the application's performance on uncompressed data without a significant gap in our system, as long as Δk is configured as previously described. Furthermore, this method applies to live consumption of data only.

Benefits. The only previous work that involves a certain degree of compression ratio adaptation is SparSDR. Based on the assumption that the signal of interest is always sparse (in the frequency domain), SparSDR applies a fixed energy threshold and filters out readings with insufficient energy. Consequently, it implicitly controls the compression ratio as it considers the activities in the spectrum. However, as discussed in § II, the sparsity assumption is not always valid, especially for spectrum measurement using low-cost sensors, which usually have a relatively narrow bandwidth. Furthermore, it completely misses the opportunity provided by the

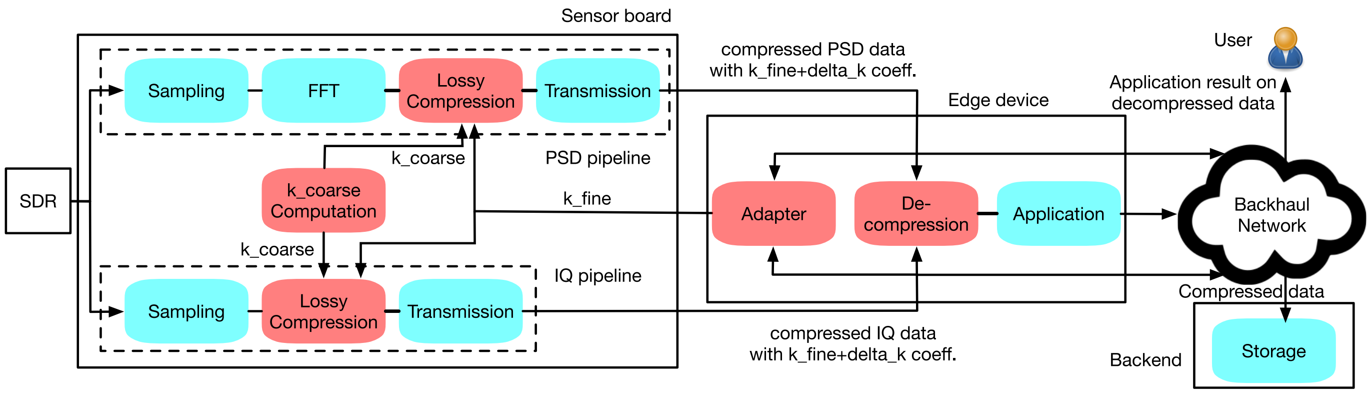


Fig. 7. Data analysis pipeline for applications on real-time spectrum data. Red blocks are FlexSpec's modules. Applications with low latency and high computational capability use the adapter deployed in an edge device. Applications on historical data run on the backend, where no adapter is needed.

nature of different applications. We take a step forward by introducing a feedback loop that considers both sides. Without additionally leveraging the nature of the application, there is not much room for compression of spectrum data that contains long-lived signals and is collected by low-cost sensors with a relatively low sampling rate. Therefore, it makes our adaptive compression generalizable to various applications and a diverse group of sensors.

Place to deploy the adapter. There are three options to deploy the adapter: (i) the sensor, (ii) the backend, (iii) and the edge device. The sensor has the lowest network latency but limited computing resources. The backend has the highest network latency but most computing resources available. Therefore, we decide to deploy the adapter on the edge device to balance the network latency and computational capability.

V. Implementation

We implement FlexSpec in a combination of C++ and python. The lossy compression algorithm is implemented in C++ for efficiency, and the logic of the compression ratio adaption scheme is implemented in python to incorporate potential applications that can leverage existing packages. Each sensor includes a 2.4M samples per second RTL-SDR radio [7] and a RBPi-4 board [5] with 4 CPU cores and 4GB memory. Fig. 7 shows the whole data analysis pipeline. The light blue blocks represent the modules that are standard spectrum crowdsensing modules, and the red blocks represent the new modules that we add to realize FlexSpec. Note that the sampling modules output IQ samples, so there is another FFT module to get the PSD readings in the PSD pipeline.

The adapter and the application run on the same device. The adapter also needs to evaluate the application performance using a few compression ratios, which may lead to significant computational overheads. As stated earlier, we chose the edge device as the place to deploy our adapter. We explored various types of edge devices deployed in the access network of the sensors and finally decided to use Intel NUC [2] due to its lower computation latency. The backend is instead responsible for managing the infrastructure, storing compressed data for later processing (c.f. § IV), and running applications with historical data.

Moreover, to make the adapter more scalable and reduce the latency of the feedback loop, we apply the following implementation tricks. After receiving a data record, we use the binary search or the simplified logic mentioned in § IV to reduce the time complexity of computations on each record. We also parallelize the computations for the m received records from each client and the computations of multiple clients by applying multi-threading. By applying these tricks, we have kept the latency of the feedback loop under the case of multiple clients on the order of 10 to 100ms, which is acceptable to capture the variation of spectrum utilization changes.

In what follows, we first benchmark the performance of FlexSpec's FWHT-based spectrum compression method in § VI, then evaluate FlexSpec's application-oriented compression ratio adaption scheme in § VII. Data for experiments are collected using RTL-SDR with 2.4 MHz complex sampling rate unless stated otherwise.

VI. Evaluation of Compression Algorithm

We first benchmark our FWHT-based spectrum data compression algorithm without any adaptation. In the study, we compare against AirPress, designed to work with PSD data, and SparSDR, designed to work with IQ data. Note that in this section, we modify SparSDR slightly to control the energy threshold to get the desired compression ratio. We also extend AirPress to be applied to IQ data for comparison.

Summary: (1) FlexSpec has up to 5 dB less reconstruction error for PSD data than Airpress because less information loss is introduced. (2) Since detailed information is maintained, we achieve up to 20% less false negative rate than Airpress, when detecting narrowband signals using PSD data. (3) We have around 70% more message decoding rate for ADS-B signal than SparSDR because its baseband waveform is similar to square waves. (4) For signal classification based on IQ data, FlexSpec achieves up to 10% more accuracy compared with SparSDR because we apply compression on I and Q readings separately.

A. Reconstruction Error for PSD Data

A common application based on PSD data is spectrum visualization using a waterfall plot. A small reconstruction

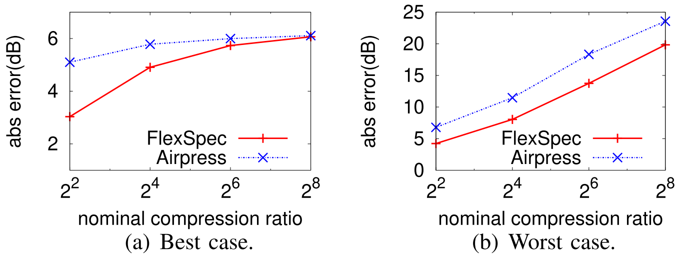


Fig. 8. 99th percentile reconstruction error as a function of nominal compression ratio. FlexSpec outperforms Airpress by up to 5 dB in both the best and the worst cases.

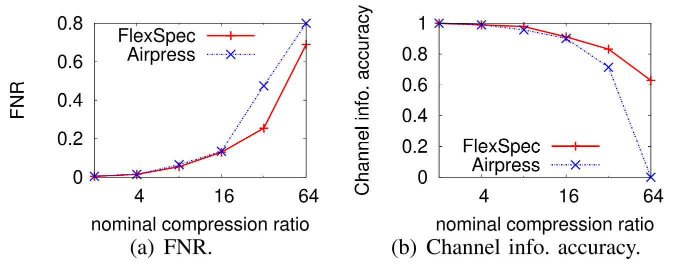


Fig. 9. Signal detection results as a function of nominal compression ratio. FlexSpec outperforms Airpress in both FNR and channel information accuracy when the compression ratio is high.

error (e.g., less than 5 dB) is visually negligible for users. We use the dataset in [48] to benchmark the reconstruction error introduced by FlexSpec's compression algorithm on PSD data for wideband measurements (from 300 MHz to 4 GHz on a 100 MHz basis). Because the dataset is large, we implement our compression method and the baseline Airpress using Spark [8] and run it in a cloud.

We first fix a compression ratio and do the compression, calculate the 99th percentile of absolute reconstruction error for all 100 MHz bands, and record the best and the worst result among all 100 MHz bands. We then vary the compression ratio to get how the best and worst case results are influenced. Fig. 8(a) and 8(b) show the best case and the worst case result, respectively. From Fig. 8(a), we can see that if the 100 MHz band is idle most of the time, the 99th percentile error is less than 6 dB when the compression ratio is no more than 256, and the increment in error decreases as the compression ratio increases. From Fig. 8(b), we notice that if the 100 MHz band is very busy, e.g., cellular bands, the 99th percentile error grows linearly as the compression ratio increases. In addition, FlexSpec's compression algorithm has up to 5dB smaller 99th percentile absolute reconstruction error for both cases than Airpress. It demonstrates that our compression method has less information loss compared with Airpress.

B. Signal Detection Based on PSD Data

We collect PSD data of 2 MHz Bluetooth (BLE) signal as well as noise using a 56 MHz sampling rate (by a customized sensor in [24] rather than RTL-SDR) and 1024-point FFT. Next, we adjust the threshold of the edge-based signal detection method [43] such that the false positive rate (FPR) and false negative rate (FNR) on the uncompressed data are zeroes, and the channel information, including bandwidth and center frequency, are correctly outputted. We then apply FlexSpec's compression method to the dataset and see how the FNR of the detection results on the reconstructed data varies as the compression ratio increases. FPR is also considered, but the result will not be shown because it stays zero (for all methods). We also check whether the channel information can still be outputted accurately (for true positive signals only). The FNR and channel information results are shown in Fig. 9, and we compare our method with Airpress.

Fig. 9(a) shows the result of FNR. We can see that when the compression ratio is relatively small (no more than 16), FlexSpec's compression method has negligible better performance. However, when the compression ratio is large (bigger than 16), our method significantly outperforms Airpress by up to 20%. Moreover, for channel information accuracy, we can achieve up to 60% more accuracy when the compression ratio is large, which is shown in Fig. 9(b). Overall, these results show that our method is more friendly for narrowband signals (2 MHz signal with 56 MHz sampling rate) than Airpress because it can preserve more details. If the same experiment is applied to wideband signals, e.g., 20 MHz WiFi signal using the same sampling rate, neither FNR nor channel information accuracy degrades much for both methods.

C. IQ Data Decoding

We first use FM radio as the analog signal example for IQ sample decoding because the human perception of audio quality tolerates the distortion introduced by the lossy compression to a certain degree. We record IQ samples of several local FM radio channels of 200 kHz bandwidth for 30-second clips. The content of the radio contains human speeches only (male and female), and we use perceptual evaluation of speech quality (PESQ) [33] to objectively evaluate the audio quality downgrade due to the lossy compression. The PESQ score ranges from -0.5 to 4.5, and a higher score indicates the decoded audio clip based on the reconstructed data is perceptually more similar to the one based on the uncompressed data. Note that when we record the IQ samples of the 200 kHz FM radio channel with a 2.4 MHz sampling rate, other active channels near the FM radio channel we are interested in, with less energy, are also captured. Therefore, the sparsity assumption in SparSDR is not fully satisfied, but the signal can still be considered a narrowband signal. Fig. 10(a) shows the PESQ score as a function of the compression ratio for FlexSpec's compression algorithm, using the number of samples per time window equals 2048. We can see that with the increasing compression ratio, the audio quality degrades gracefully, and the audio quality becomes unacceptable (PESQ score < 1) when the compression ratio is 64. For comparison, if we extend Airpress to IQ samples, the PESQ score is less than 1 when the compression ratio is only 2, which is very close to the score of our method when the compression ratio is 64. The reason for Airpress' relatively poor performance is that details are important for FM radio decoding. Moreover, Fig. 10(a) also shows that SparSDR has a slightly higher score when the compression ratio is larger than 8.

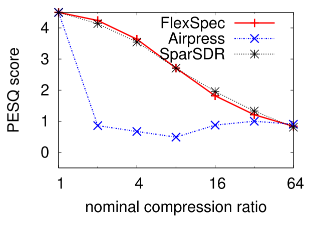
We then study the ADS-B signal. A representative PSD has already been shown in Fig. 2. ADS-B works at a carrier centered at 1090 MHz, and we study it as the digital signal example for IQ sample decoding. For decoding it, we use *dump1090* [1]. We consider the message decoding rate as the performance metric, and Fig. 10(b) shows the results. From Fig. 10(b), we find that compressed data using Airpress cannot be successfully decoded at all because the detailed information is lost. Additionally, when the compression ratio is larger than 16, all methods have zero successful rates. It is worse than FM IQ sample decoding since ADS-B message decoding tolerates less information loss than the human perception of audio quality. Moreover, compared with SparSDR, FlexSpec outperforms by up to 70% because the ADS-B signal is pulse-position modulated so that its baseband waveform is similar to rectangular waves, which can be captured more accurately by the bases of FWHT compared with that of FFT. The preambles are easier to be detected when compressed using FlexSpec. It can be observed in the experiment in Fig. 10(c), where it is evident that FlexSpec closely represents the original signal. However, prior methods largely distort it using a similar compression ratio. Finally, since the ADS-B signal is amplitude modulated only, if we apply our compression method to the amplitude presentation of the signal (FlexSpec$_p$ in Fig. 10(b)), we can have 70% more decoded messages when the overall compression ratio is 4. It is achieved by applying the compression ratio of 2 on amplitude readings, and the phase readings are omitted. Note that this is generally impossible since we usually do not know what kinds of signals are captured. The benefit in this specific case comes with 19% more time overhead due to converting between IQ coordinates and amplitude/phase coordinates.

We also compare the performance of FlexSpec with that of SparSDR when the sparsity assumption is fully met. We obtained the traces of the BLE signal from SparSDR's authors. These traces only capture a 2 MHz BLE signal, which is frequency modulated, using a 100 MHz sampling rate. We found that the decoding rate performance of FlexSpec is significantly worse than SparSDR (roughly 25% vs. 95%). However, this case happens rarely with low-cost sensors due to their limited sampling rate compared with high-end sensors, e.g., USRP. We further downsample the traces to 10 MHz to obtain a moderate sparsity and then inject white noise of various levels. Fig. 10(d) shows the decoding performance as a function of added noise power. The normalized added noise of 0 dB stands for the noise power level that the decoding performance starts to degrade for both methods. We can see that with frequency modulated BLE signals and a moderate sparsity, the decoding performance using FlexSpec and SparSDR compressed data are close regardless of the noise level. In addition, it would be interesting to explore a hybrid method to combine SparSDR with FlexSpec methods for extremely sparse signals in the future.

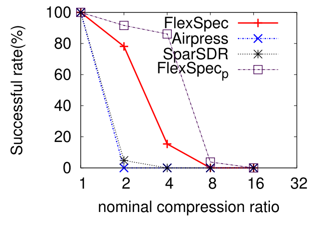
D. Signal Classification Based on IQ Data

IQ samples can be used to classify modulation schemes hence the types of signals being transferred, e.g., WiFi or LTE. We use the deep learning model described in [31] and the IQ dataset in [3] to classify whether the IQ segments (of length 128) are WiFi or LTE signals and evaluate how FlexSpec's compression algorithm affects the classification accuracy. We apply the compression algorithm to the training data and feed the reconstructed training data to the model, then vary the compression ratio and see how the accuracy of the classifier on the reconstructed training dataset changes. Fig. 11(a) compares our performance with those using (extended version of) Airpress and SparSDR. Note that the sparsity assumption in SparSDR is not met because the signal's bandwidth is equal to the sampling rate. From Fig. 11(a), we can maintain more than 90% accuracy when the compression ratio is no more than 16. On the contrary, the accuracy of Airpress directly drops to 50%, i.e., the accuracy of random guessing, when the compression ratio is 4. Moreover, when the compression ratio is no less than 16, our method's accuracy outperforms SparSDR by up to 15%.

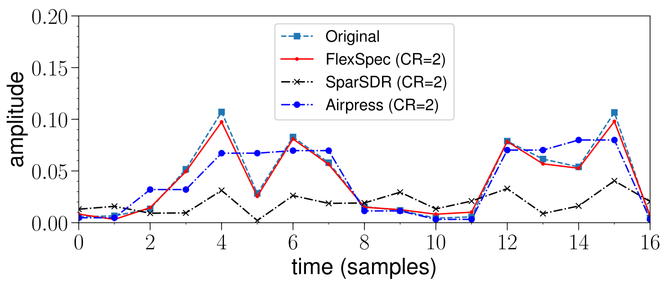
We then investigate whether this performance increment is due to FWHT's better performance over FFT or the fact that we apply the FWHT-based compression algorithm on I and Q readings separately. We compare our method with the one that



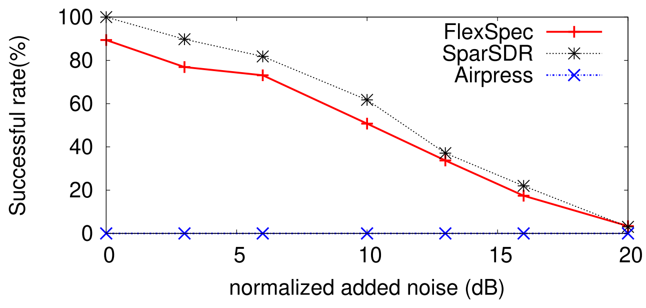
(a) FM audio quality.



(b) ADS-B signal decoding rate.

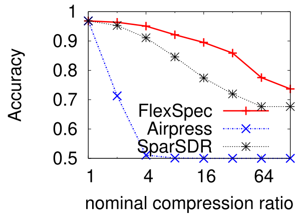


(c) ADS-B preamble reconstruction in the time domain.

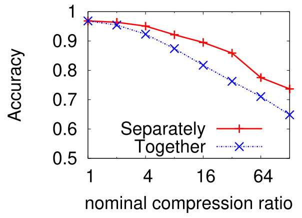


(d) BLE signal decoding result with moderated sparsity and injected noise.

Fig. 10. IQ sample decoding results. Above: when the sparsity assumption is not met, FlexSpec approximates the original signal, while prior methods largely distort it. Below: when the sparsity assumption is fully met and the signal is frequency modulated, SparSDR outperforms FlexSpec.



(a) Comparison with baselines.



(b) Apply compression on IQ readings together vs. separately.

Fig. 11. Classification accuracy results as a function of nominal compression ratio. FlexSpec outperforms both Airpress and SparSDR. The performance enhancement over SparSDR mainly comes from compression on I and Q readings separately.

applies (complex) FWHT-based compression on IQ readings together. In other words, the relationship between $H_{2^{n+1}}$ and H_{2^n} in Eq. (1) is modified as [16]:

$$H_{2^{n+1}} = \begin{bmatrix} H_{2^n} & i \cdot H_{2^n} \\ -i \cdot H_{2^n} & -H_{2^n} \end{bmatrix}, \quad n = 1, 2, 3, \ldots, \tag{4}$$

and the initial condition is modified as $H_2 = \begin{bmatrix} 1 & i \\ -i & -1 \end{bmatrix}$. Fig. 11(b) shows the result, and we can see that if the compression is applied on I and Q readings together, it has up to 10% accuracy drop than applying compression on I and Q readings separately. Therefore, the accuracy gain of our method is attributed to the fact that we apply the FWHT-based compression algorithm on I and Q readings separately. We conclude that the deep learning model to classify signal exploits the I and Q differently, so it is beneficial to compress them separately.

E. Correlation Reduction and Delay

Some applications, e.g., IQ localization based on Time Differential of Arrival, require stringent delay requirements and a strong correlation with the uncompressed data. In this regard, any information loss would immediately result in lower application performance, hence is challenging if compression is considered. We investigate the delay and reduction of correlation introduced by FlexSpec compression for ADS-B and IoT-433 signals in Fig. 12. IoT-433 signals are signals from IoT devices at ISM 433 MHz band with 0.1 MHz bandwidth for most of the signals, mainly for reporting information, e.g., temperature, humidity, presence, or even the Tire Pressure Monitoring System (TPMS) that vehicles use. We use ADS-B and IoT-433 signals as exemplary wideband and narrowband signals. In the experiments, we use a single antenna and a splitter that connects the signal to the two sensors. The results show that the correlation decreases as soon as we compress the signals as expected. Despite this, we can achieve acceptable delays between the compressed and uncompressed data up to a compression ratio of 4.

VII. Evaluation of Compression Adaptation

As explained in previous sections, the compression ratio can be selected based on the performance of the specific application running in the backend that consumes the compressed spectrum data. In this section, we evaluate

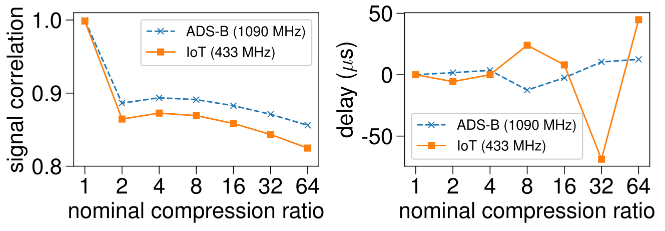


Fig. 12. Signal correlation and delay vs. nominal compression ratio for both IoT-433 and ADS-B. Up to compression factor of 4 the delay is acceptable for localization apps.

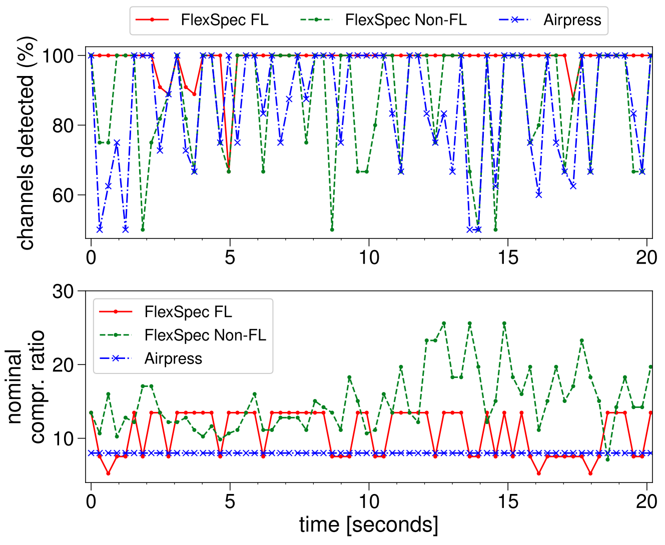


Fig. 13. Channel detection over compressed PSD data. Application-oriented compression ratio adaptation with Feedback Loop performs better than generic approaches.

FlexSpec's application-oriented compression ratio adaptation scheme using two real scenarios, which are (i) channel detection based on PSD data, and (ii) message decoding using IQ data. Note that Δk is $1/16$ of the segment size in all experiments.

Summary: (1) For detecting the number of FM channels based on PSD data, if we target optimizing the difference of the number of detected channels to be smaller than 1, only FlexSpec (with feedback loop) can achieve it. (2) Compared with no adaptation in Airpress, we can achieve up to 7× more uplink data size reduction when the spectrum changes to idle. (3) We use both wideband ADS-B and narrowband IoT-433 signals for IQ decoding application. (4) reduces messages unable to be decoded by 6× to 8× while reducing uplink data size even more, compared with SparSDR using a power threshold that gives a similar nominal compression ratio.

A. Feedback Loop

In this experiment, we evaluate the performance of channel detection over an FM radio band, which contains 9 active channels with varying energy over time, using compressed PSD data. We compare the performance of FlexSpec with and without *Feedback Loop* (FL) and also compare them with Airpress. Fig. 13 (top) shows the channels detected over PSD

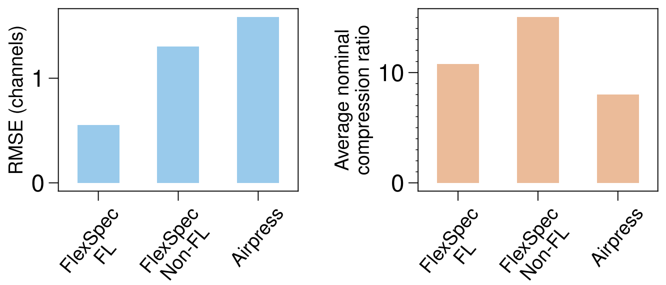


Fig. 14. RMSE of channels detected (left) and average nominal compression ratio for the different solutions evaluated over PSD data.

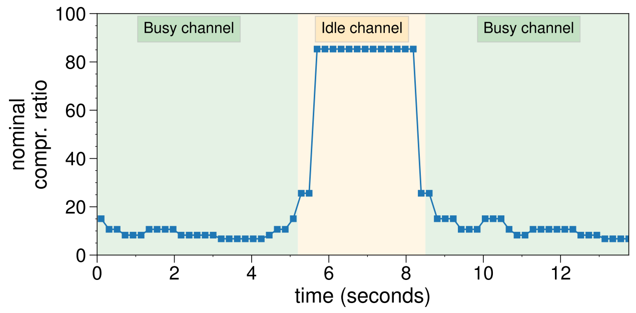


Fig. 15. Compression ratio is set adaptively in a busy-idle channel scenario to reduce data that are backhauled. Compression ratio adapted every 200 ms.

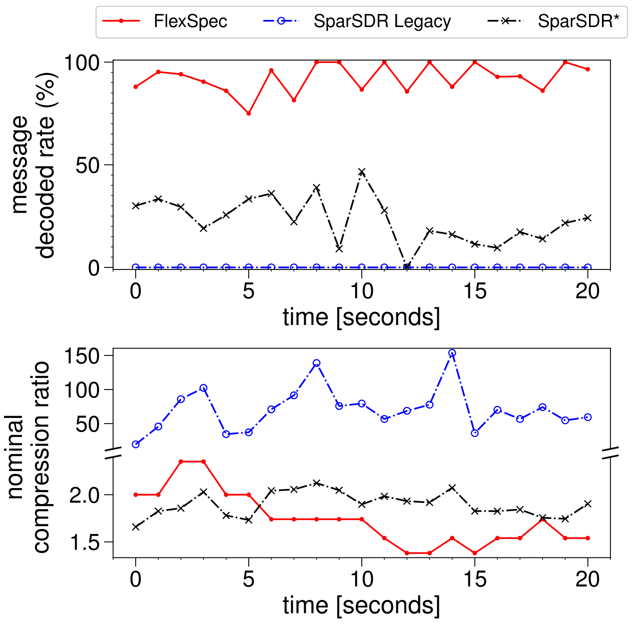


Fig. 16. ADS-B messages decoded and compression ratio used for FlexSpec and SparSDR.

data for different methods. FlexSpec FL adapts the compression ratio dynamically so that the difference in the number of detected channels smaller than 1 is tolerated. FlexSpec Non-FL determines the new compression ratio by executing a simple application on the sensor side (to not overload it). It compresses the original signal until the reconstructed one introduces at most a 5 dB error. This process is executed on the sensor side with no FL involved. For a fair comparison with Airpress, we select a nominal compression ratio (8) that gives a data compression ratio similar to the average of FlexSpec FL. Fig. 14 shows that FlexSpecFL performs near perfect in terms of channels detected while keeping the average compression ratio about 10. FlexSpec Non-FL reports a higher nominal compression ratio but also introduces an RMSE of the channels detected larger than 1. It indicates that simple adaptation running on the sensor without involving FlexSpec's FL does not optimize applications' performance to get the tolerable performance loss. Lastly, Airpress, which used a fixed compression ratio, reports an even higher RMSE of channels detected, which shows adaptation is needed even in a relatively stable band.

B. Busy-Idle State

We evaluate how the nominal compression ratio is set depending on the state of the spectrum and the application running in the edge device. In this experiment, we collect PSD data on the same FM radio band in the previous setting. In order to simulate the idle state, we disconnect the antenna from the radio receiver. Fig. 15 shows how the nominal compression ratio is set adaptively. We notice that during the idle channel state, FlexSpec increases the compression ratio (by decreasing k_{fine}) since there is no useful signal information to detect channels, reducing the size of uplink data. We achieve up to 10× more nominal compression ratio during the idle state than Airpress, which uses a fixed compression ratio (8), as Fig. 13 shows. Considering the overhead of data format for FlexSpec and Airpress, this translates to up to 7× more reduction in uplink data size..

C. IQ Decoding

We evaluate FlexSpec over IQ data with the goal of maximizing the message decoded rate. For this experiment, we use ADS-B and IoT-433 signals as the example wideband and narrowband signals, respectively.

Fig. 16 compares FlexSpec and SparSDR regarding their message decoded rate for ADS-B. For the ADS-B signal, the compression is performed in segments of 2k samples, and the message decoding is performed in the adapter every 1M samples using *dump1090* [1]. The latter also means that for every 1M samples, FlexSpec decides on the new compression ratio to be used. For SparSDR legacy, we use a power threshold of 5 dB above the noise floor as suggested, but we find no message can be decoded. Therefore, we also tune the power threshold such that SparSDR outputs a similar nominal compression ratio as FlexSpec, denoted as SparSDR*. Fig. 18 shows that for ADS-B, FlexSpec has 6× fewer messages unable to be decoded compared with SparSDR*, at a very similar nominal compression ratio. Considering the data format of FlexSpec and SparSDR and the overlapping time window in SparSDR, FlexSpec's 6× fewer messages unable to be decoded comes with a 1.7× more reduction in uplink data size.

For the IoT-433 signal, we apply the same configuration as ADS-B, but the *rtl_433* decoder [6] is used. Fig. 17 shows how FlexSpec outperforms SparSDR. Similar to ADS-B, if we use a power threshold of 5 dB above the noise floor (SparSDR

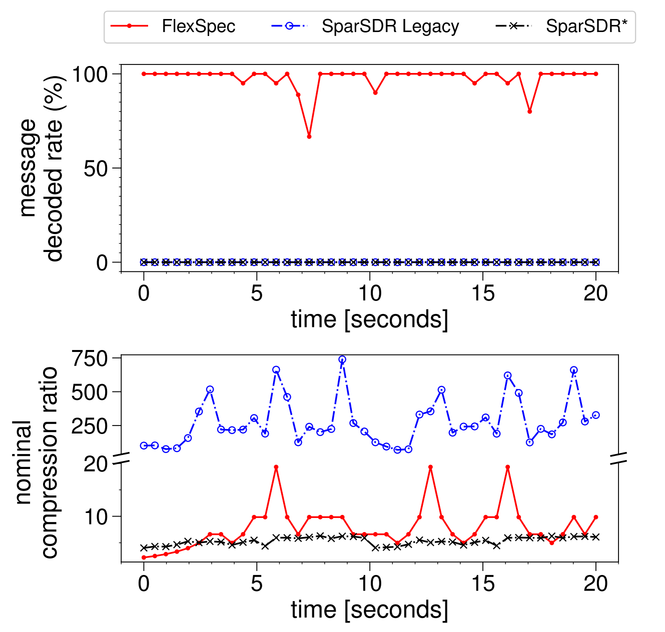


Fig. 17. IoT-433 messages decoded and compression ratio used for FlexSpec and SparSDR.

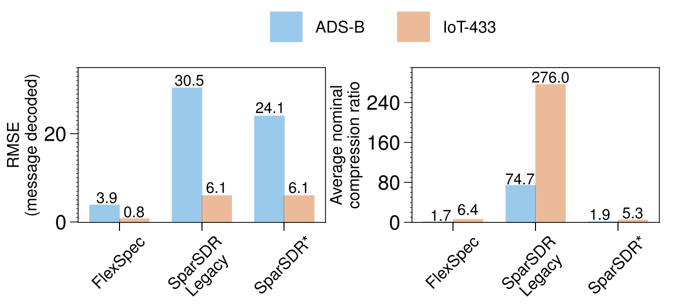


Fig. 18. Summary of IQ decoding performance and compression ratio using FlexSpec and SparSDR.

legacy), no messages can be decoded. However, different from ADS-B, if we apply a power threshold that gives a similar nominal compression ratio to FlexSpec (SparSDR*), a significant number of messages still cannot be decoded. Fig. 18 shows that, for IoT-433 signal, FlexSpec has 8× fewer messages undecodable compared with SparSDR*, with a 1.2× more nominal compression ratio. If the data format of FlexSpec and SparSDR and the overlapping time window in SparSDR are considered, we have a 2.2× more reduction in uplink data size.

D. Latency Performance

The performance of our solution based on FWHT does not depend on the compression coefficient used but on the segment size of compressed/decompressed data. We evaluate the performance of the compression algorithm in different platforms (RBPi-4 and Intel NUC) for different segment sizes. Fig. 19 shows how the (de)compression processing time grows exponentially with the segment size. Due to more computational resources, the compression/decompression process is 16× faster on the Intel NUC than on the sensor (RBPi-4). However, running compression/decompression on the sensor is still within several milliseconds. For the reference segment size (2048) used in all the evaluations in this section, compress/decompress functions take 0.20 ms in the edge device. Additionally, although we evaluate latency performance on spectrum data obtained from RTL-SDR, FWHT can also be computed on sensors with higher sampling rates. The reason is that, in that case, there is still an FFT module in the PSD pipeline, as shown in Fig. 7. As long as FFT can be computed on the sensor, so can FWHT, as FWHT has a similar complexity as FFT (§ III).

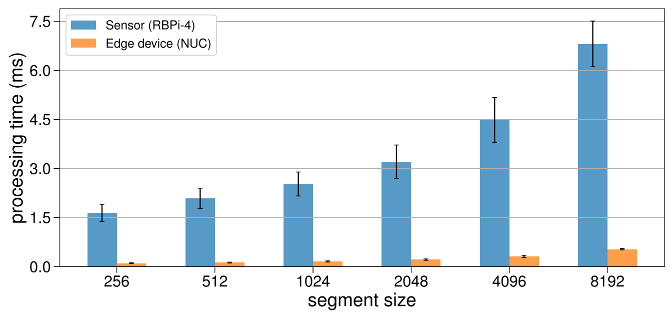


Fig. 19. FlexSpec compress/decompress latency performance in sensor (RBPi-4) and edge device (Intel NUC).

TABLE I
LATENCY OF FLEXSPEC'S FEEDBACK LOOP FOR DIFFERENT STREAMING APPLICATIONS EVALUATED ON BOTH SCENARIOS: RBPI-4 AND RBPI-4 + INTEL NUC

Pipeline App	RBPi-4 (ms)	RBPi-4 + Intel NUC (ms)
Channel detection (PSD)	144	37
dump1090 (IQ)	264	115
IoT433 (IQ)	1080	220

We also evaluate the latency of FlexSpec in terms of how long it takes to provide a new compression coefficient according to the application executed on decompressed data. Table I shows the results of the two different scenarios evaluated. In the first scenario (named "RBPi-4"), compressing/decompressing data and streaming applications are executed on the sensor. The second one (named "RBPi-4 + Intel NUC") performs the compression on the sensor, but decompression and streaming applications are executed on the edge device (Intel NUC). Even assuming a transmission delay of 5 ms between the sensor and the edge device, we obtain a lower latency in the "RBPi-4 + Intel NUC" scenario, meaning that we can provide an optimal compression coefficient more often (on average 3.5× faster). Moreover, the final latency of FlexSpec's feedback loop strongly depends on the streaming application executed to obtain the optimal compression coefficient.

VIII. DISCUSSION AND LIMITATIONS

Hard limit on uplink bandwidth. There can be a hard limit on a sensor's uplink bandwidth from the backhaul network, and this limit may vary for different sensors. For instance, sensors with a mobile connection to the backend or deployed in developing regions may have more stringent constraints. Incorporating this limit to our application-oriented

TABLE II
HIGH-LEVEL COMPARISON WITH THE STATE-OF-THE-ART

	FlexSpec	Airpress [51]	SparSDR [22]	Gzip [29]	BigSpec [48]	SAIFE [32]
data type	PSD & IQ	PSD only	IQ only	general	PSD & IQ	PSD & IQ
real-time support	Yes	Yes	Yes	Yes	No	No
explicit adaption	Yes	No	No	No	via re-training	via re-training
hardware requirement	low-cost sensors	low-cost sensors	low-cost sensors	low-cost sensors	high-end sensors & servers	GPU/high-end sensors

compression ratio adaption by simply limiting the highest compression coefficient used by the sensor is not enough. We need to also consider the overhead of data schema, sparse encoding, etc. In addition, the application's performance may be unsatisfactory if the hard limit is too aggressive, and the degree of dissatisfaction depends on the nature of the application. When there is a conflict between the application's performance and the hard limit on the uplink bandwidth, it is the user's decision whether to favor the uplink bandwidth or the application, and it should be done on a case-by-case basis.

Privacy. Collecting spectrum data has multiple privacy implications. It is subject to government regulations on storing spectrum data abroad and for which purposes. Metrics are needed to measure the privacy risk of sending a certain amount of spectrum data. FlexSpec could be applied to transmit limited information abroad, smaller than what is transmitted in the country of origin, by tuning the compression coefficient.

Age of information. Age of information is a concept that has achieved great attention in the research community in the last years [14]. It measures the freshness of the state of the source, and its application to the problem of application-oriented spectrum compression is so far unexplored. FlexSpec could be extended to verify how long the sensor can rely on the compression ratio inferred by the adapter.

Human-in-the-loop labeling. FlexSpec is not designed for applications that need human-in-the-loop to grade applications' performance. However, suppose only a small amount of data needs to be evaluated by humans. In that case, it is possible to extend FlexSpec with some crowdsourcing application performance evaluation function to make it support this kind of application. The trade-off is that it leads to a higher latency in deciding the compression ratio at the sensor.

Features extractions and decoding in the sensor. There are two potential alternatives to performing data compression. The first one is extracting large spatio-temporal scale features in the spectrum data. However, they cannot be extracted from the sensor in real-time unless we relax the low time and space complexity requirements presented in § II. Given the current hardware costs, the backend is a much better fit for this processing. The second one is data decoding in the spectrum sensor, e.g. [10], instead of doing it in the backend or the edge device, e.g. [15], assuming the computational cost is tolerable. It is left as future work an extensive evaluation of which decoders can run in the sensor and which cannot (for instance, srsLTE for LTE decoder requires FPGA implementation for real-time decoding, which cannot be afforded by the single low-cost sensor [17]). Furthermore, as the backend still requires data for historical access, we still need to access compressed data in the backend.

IX. RELATED WORK

Spectrum crowdsensing: Spectrum measurement has been traditionally done using commercial spectrum analyzers [4], [21], [37], [44], [46], [49], which is both bulky and costly so that large-scale spectrum measurement is hardly possible. Recently, with the cost of low-end sensor keeps decreasing [18], [21], [24], [27], [50], crowdsensing with low-cost sensors becomes a popular paradigm [13], [26], [30], [34], [36], [39], which enables very large-scale measurement. In addition, crowdsensing recently has been used to not only sense the spectrum but also decode the spectrum data [11], [12].

Spectrum data compression: Overall, Table II gives the high-level comparison of FlexSpec with state-of-the-art spectrum data compression methods in terms of features and requirements. Electrosense [29], [30] only leverages lossless Gzip compression to pack data into bits compactly. However, it does not reduce data size much due to noise (about 10% based on our experience). BigSpec [48] utilizes truncated SVD to do data compression on a large amount of spectrum data, but it works only for (historical) batch data analysis rather than streaming data. Similarly, autoencoders can compress batch data but need a significant amount of time for training before one can obtain the model for compression [32]. For this reason, they require powerful machines (sensors and/or servers). Airpress [51] and SparSDR [22] are the most closely related works to ours because they also focus on compressing streaming PSD and IQ data, respectively We have performed an extensive comparison against them in this work. They only propose a compression method with a fixed/unadjustable compression ratio for the specific data format. On the other hand, we propose a compression method that works for both PSD and IQ data. FlexSpec also offers an application-oriented compression ratio adaptation scheme to send as little data as possible between the sensor and the edge device it connects to, and fulfills the need of different applications' performance requirements.

FWHT: Although we are the first to apply FWHT to spectrum data compression, FWHT-based compression has been widely applied to other fields. Traditional fields of FWHT-based compression include image compression, e.g., JPEG XR [42], and video compression, e.g., MPEG-4 [23]. It has recently been adopted into IoT data compression, e.g., weather data [25]. FWHT is also extensively utilized in applications other than compression. For instance, CDMA systems adopt FWHT to solve multi-access and interference cancellation problems to increase channel capacity [40], [41].

Feedback loop-based rate adaptation: The idea of rate adaptation based on feedback loops is classic and has been successfully applied in different areas. One example is to use a feedback loop to control the video rate in wireless scenarios, e.g., QFlow [9], which uses QoE metrics as the feedback. Another example is rate adaptation at the MAC layer for the 802.11 networks [45], which usually uses throughput or frame successful rate as the feedback. Our application-oriented compression ratio adaptation scheme shares the same concept. However, FlexSpec differs from other rate adaptation schemes because it mainly depends on two unique factors: i) information embedded in the spectrum data; ii) application related to spectrum data selected by the end-user.

X. Conclusion

A key hurdle to building multiple spectrum applications with ubiquitous spectrum crowdsensing systems is the high uplink bandwidth needed and the resulting demands on centralized storage. We presented FlexSpec, a general adaptive uplink data compression framework, which has the objective of removing this hurdle. FlexSpec solves the main drawbacks of prior work, namely the problem of missing critical information in the spectrum compression technique and lack of adaptation for the application needs. We have experimentally shown the benefits of our framework with several spectrum applications, from signal detection to IQ decoding. We envision that FlexSpec's techniques will be an integral component of large spectrum crowdsensing systems. The datasets used in this paper can be accessed at https://repository.electrosense.org/datasets/tnet_2021/

References

[1] *Dump1090*. Accessed: Oct. 15, 2021. [Online]. Available: https://github.com/openskynetwork/dump1090-hptoa

[2] *Intel Nuc*. Accessed: Oct. 15, 2021. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/nuc.html

[3] *LTE-WiFi-IQ-Samples*. Accessed: Oct. 15, 2021. [Online]. Available: https://github.com/ewine-project/lte-wifi-iq-samples

[4] *Microsoft Spectrum Observatory*. Accessed: Oct. 15, 2021. [Online]. Available: https://observatory.microsoftspectrum.com/

[5] *Raspberry PI 4*. Accessed: Oct. 15, 2021. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[6] *RTL-433*. Accessed: Oct. 15, 2021. [Online]. Available: https://github.com/merbanan/rtl_433

[7] *RTL-SDR*. Accessed: Oct. 15, 2021. [Online]. Available: https://www.rtl-sdr.com

[8] *Spark*. Accessed: Oct. 15, 2021. [Online]. Available: https://databricks.com/spark/

[9] R. Bhattacharyya et al., "QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 251–260.

[10] R. Calvo-Palomino et al., "Electrosense+: Crowdsourcing radio spectrum decoding using IoT receivers," *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107231.

[11] R. Calvo-Palomino, H. Cordobés, F. Ricciato, D. Giustiniano, and V. Lenders, "Collaborative wideband signal decoding using non-coherent receivers," in *Proc. 18th Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2019, pp. 1–9.

[12] R. Calvo-Palomino, D. Giustiniano, V. Lenders, and A. Fakhreddine, "Crowdsourcing spectrum data decoding," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[13] A. Chakraborty and S. R. Das, "Designing a cloud-based infrastructure for spectrum sensing: A case study for indoor spaces," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2016, pp. 17–24.

[14] J. P. Champati, M. H. Mamduhi, K. H. Johansson, and J. Gross, "Performance characterization using AoI in a single-loop networked control system," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 197–203.

[15] A. Dongare et al., "Charm: Exploiting geographical diversity through coherent combining in low-power wide-area networks," in *Proc. 17th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2018, pp. 60–71.

[16] B. J. Falkowski and S. Rahardja, "Properties and applications of unified complex Hadamard transforms," in *Proc. 27th Int. Symp. Multiple Valued Log.*, May 1997, pp. 131–136.

[17] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. 10th ACM Int. Workshop Wireless Netw. Testbeds, Experim. Eval., Characterization*, Oct. 2016, pp. 25–32.

[18] Y. Guddeti, R. Subbaraman, M. Khazraee, A. Schulman, and D. Bharadia, "Sweepsense: Sensing 5 GHz in 5 milliseconds with low-cost radios," in *Proc. NSDI*, 2019, pp. 317–330.

[19] H. F. Harmuth, *Transmission of Information by Orthogonal Functions*. Cham, Switzerland: Springer, 2012.

[20] C. A. R. Hoare, "Algorithm 65: Find," *Commun. ACM*, vol. 4, no. 7, pp. 321–322, Jul. 1961.

[21] M. Hoyhtya et al., "Measurements and analysis of spectrum occupancy in the 2.3–2.4 GHz band in Finland and Chicago," in *Proc. 9th Int. Conf. Cognit. Radio Oriented Wireless Netw.*, 2014, pp. 95–101.

[22] M. Khazraee et al., "SparSDR: Sparsity-proportional backhaul and compute for SDRs," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2019, pp. 391–403.

[23] S.-K. Kwon, A. Tamhankar, and K. Rao, "Overview of H. 264/MPEG-4 part 10," *J. Vis. Commun. Image Represent.*, vol. 17, no. 2, pp. 186–216, 2006.

[24] Y. Li, Y. Zeng, and S. Banerjee, "Enabling wideband, mobile spectrum sensing through onboard heterogeneous computing," in *Proc. 22nd Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2021, pp. 85–91.

[25] A. Moon, J. Kim, J. Zhang, and S. W. Son, "Lossy compression on IoT big data by exploiting spatiotemporal correlation," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2017, pp. 1–7.

[26] A. Nika et al., "Empirical validation of commodity spectrum monitoring," in *Proc. 14th ACM Conf. Embedded Netw. Sensor Syst. (CD-ROM)*, Nov. 2016, pp. 96–108.

[27] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng, "Towards commoditized real-time spectrum monitoring," in *Proc. 1st ACM Workshop Hot Topics Wireless*, Sep. 2014, pp. 25–30.

[28] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[29] D. Pfammatter, D. Giustiniano, and V. Lenders, "A software-defined sensor architecture for large-scale wideband spectrum monitoring," in *Proc. 14th Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2015, pp. 71–82.

[30] S. Rajendran et al., "Electrosense: Open and big spectrum data," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 210–217, Jan. 2018.

[31] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018.

[32] S. Rajendran, W. Meert, V. Lenders, and S. Pollin, "SAIFE: Unsupervised wireless spectrum anomaly detection with interpretable features," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Oct. 2018, pp. 1–9.

[33] *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, document Rec. ITU-T P. 862, Recommendation, I.-T, 2001.

[34] B. Reynders et al., "SkySense: Terrestrial and aerial spectrum use analysed using lightweight sensing technology with weather balloons," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 352–363.

[35] T. Ritter. (1996). *Walsh-Hadamard Transforms: A Literature Survey. Research Comments From Cipers by Ritter*. [Online]. Available: www.ciphersbyritter.com/RES/WALHAD.HTM

[36] A. Saeed, K. A. Harras, E. Zegura, and M. Ammar, "Local and low-cost white space detection," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 503–516.

[37] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu, "Snow: Sensor network over white spaces," in *Proc. Int. Conf. Embedded Netw. Sensor Syst. (ACM SenSys)*, 2016, pp. 272–285.

[38] *Spectrum Occupancy Measurements and Evaluation Report*, document ITU-R SM.2256-1, Series, S, 2018.

[39] J. Shi, Z. Guan, C. Qiao, T. Melodia, D. Koutsonikolas, and G. Challen, "Crowdsourcing access network spectrum allocation using smartphones," in *Proc. 13th ACM Workshop Hot Topics Netw.*, Oct. 2014, p. 17.

[40] X. Tang and W. H. Mow, "Design of spreading codes for quasi-synchronous CDMA with intercell interference," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 1, pp. 84–93, Jan. 2006.

[41] S.-H. S. Tsai, "Multicode transmission using Walsh Hadamard transform," U.S. Patent 7 869 546, Jan. 11, 2011.

[42] C. Tu, S. Srinivasan, G. J. Sullivan, S. Regunathan, and H. S. Malvar, "Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR/HD photo," *Proc. SPIE*, vol. 7073, Sep. 2008, Art. no. 70730C.

[43] L. Yang, W. Hou, L. Cao, B. Y. Zhao, and H. Zheng, "Supporting demanding wireless applications with frequency-agile radios," in *Proc. NSDI*, 2010, pp. 65–80.

[44] S. Yin, D. Chen, Q. Zhang, M. Liu, and S. Li, "Mining spectrum usage data: A large-scale spectrum measurement study," *IEEE Trans. Mobile Comput.*, vol. 11, no. 6, pp. 1033–1046, Jun. 2012.

[45] W. Yin, P. Hu, J. Indulska, M. Portmann, and Y. Mao, "MAC-layer rate control for 802.11 networks: A survey," *Wireless Netw.*, vol. 26, no. 5, pp. 3793–3830, Jul. 2020.

[46] X. Ying, J. Zhang, L. Yan, G. Zhang, M. Chen, and R. Chandra, "Exploring indoor white spaces in metropolises," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2013, pp. 255–266.

[47] Y. Zeng, R. Calvo-Palomino, D. Giustiniano, G. Bovet, and S. Banerjee, "Adaptive uplink data compression in spectrum crowdsensing systems," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Dec. 2021, pp. 114–122.

[48] Y. Zeng, V. Chandrasekaran, S. Banerjee, and D. Giustiniano, "A framework for analyzing spectrum characteristics in large spatio-temporal scales," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2019, pp. 1–16.

[49] T. Zhang, N. Leng, and S. Banerjee, "A vehicle-based measurement framework for enhancing whitespace spectrum databases," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2014, pp. 17–28.

[50] T. Zhang, A. Patro, N. Leng, and S. Banerjee, "A wireless spectrum analyzer in your pocket," in *Proc. 16th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2015, pp. 69–74.

[51] M. Zheleva, T. Larock, P. Schmitt, and P. Bogdanov, "AirPress: High-accuracy spectrum summarization using compressed scans," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Oct. 2018, pp. 1–5.

Yijing Zeng (Student Member, IEEE) received the bachelor's degree in communication engineering from the Beijing University of Posts and Telecommunications in 2008, the master's degree in electrical and computer engineering from the University of California at Santa Barbara in 2015, and the Ph.D. degree in computer sciences from the University of Wisconsin–Madison in 2022. His current research interests include building large-scale spectrum sensing and data analysis platforms.

Roberto Calvo-Palomino (Member, IEEE) received the Ph.D. degree in telematics from the University Carlos III of Madrid and the IMDEA Networks Institute, Madrid, Spain, July 2019. He is currently an Associate Professor with Rey Juan Carlos University, Madrid. He also made a post-doctoral research at IMDEA Networks. He worked as a Senior Software Engineer for five years in the industry. His main research interests include the IoT, data analysis, collaborative-smart systems deployed at large scale, collaborative/distributed algorithms to build smart crowd-sourcing platforms, collaborative wideband spectrum monitoring, anomaly detection, and fully distributed systems on a large scale.

Domenico Giustiniano (Senior Member, IEEE) received the Ph.D. degree in telecommunication engineering from the University of Rome Tor Vergata in 2008. He is currently a Research Associate Professor (tenured) with IMDEA Networks, Madrid, Spain. Before joining IMDEA, he was a Senior Researcher and a Lecturer at ETH Zürich. He also worked for a total of four years as a Post-Doctoral Researcher at Industrial Research Laboratories (Disney Research, Zürich, and Telefónica, Research Barcelona). He is also the Leader of the OpenVLC Project and the Co-Founder of the non-profit Electrosense Association. He has authored over 100 international papers. His current research interests include the battery-free IoT, large-scale spectrum analytics, and 5G+ localization systems.

Gerome Bovet received the Ph.D. degree in networks and systems from Telecom ParisTech, France, in 2015, and the Executive M.B.A. degree from the University of Fribourg, Switzerland, in 2021. He is currently the Head of data science with Armasuisse, Switzerland, where he leads a research team and a portfolio of about 30 projects. His work focuses on machine and deep learning approaches applied to cyber-defense use cases, with an emphasis on anomaly detection, adversarial and collaborative learning.

Suman Banerjee (Fellow, IEEE) received the bachelor's degree from IIT Kanpur, and the M.S. and Ph.D. degrees from the University of Maryland. He is currently the David J. DeWitt Professor of computer sciences with the University of Wisconsin–Madison and the Founding Director of the WiNGS Laboratory, which broadly focuses on research in wireless and mobile networking systems. He has published more than 100 technical papers in leading conferences and journals. He is a Fellow of the ACM. He was an inaugural recipient of the ACM SIGMOBILE Rockstar Award and a recipient of the U.S. National Science Foundation Career Award. He served as the Chair for ACM SIGMOBILE from 2013 to 2017.