Improving Network Resource Utilization for Distributed Wireless Sensing Applications

Snehadeep Gayen, Yamini Shankar and Ayon Chakraborty

SENSE Lab, IIT Madras

ABSTRACT

Edge-assisted wireless sensing is increasingly popular, where complex neural network models perform inference tasks on wireless channel state information (CSI) data streamed from IoT devices. However large volumes of CSI data sent across the network for inference can significantly impact network bandwidth and reduce the Quality of Experience. This paper tackles the challenge of optimizing network resource utilization in wireless sensing systems by compressing and subsampling CSI streams. We evaluate methods that quantize and selectively subsample CSI data before transmission to the edge server, which is then fed to the inference models. Such approach reduces bandwidth and computational load, improving data transmission and processing efficiency. Experiments conducted in two real testbeds (indoors as well as outdoors) show how CSI compression preserves sensing information integrity while enhancing system performance in terms of latency, energy efficiency, and throughput. By integrating quantization and subsampling with edge computing, this work enhances wireless sensing systems, making them more scalable and efficient in utilizing network resources.

CCS CONCEPTS

• **Computer systems organization** → Sensor networks; **Real-time system architecture**; • **Human-centered computing** → **Empirical studies in ubiquitous and mobile computing**.

KEYWORDS

Wireless Sensing, Channel State Information, Compression

ACM Reference Format:

Snehadeep Gayen, Yamini Shankar and Ayon Chakraborty . 2024. Improving

Network Resource Utilization for Distributed Wireless Sensing Applications. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Internet of Things (IoT) devices, majority of which are equipped with wireless radios, offer a unique opportunity to leverage their capabilities for Radio Frequency (RF)-based sensing. RF-based sensing (a.k.a. wireless sensing) generally refers to a host of techniques that utilize the underlying wireless communication channel as a sensing modality. For instance, the Channel State Information (CSI) available at a Wi-Fi receiver includes amplitude and phase variations of the received signal, which are perturbed by the surrounding physical environment and the movement of objects or reflectors (e.g., human beings) within it. Analyzing such variations allows for learning and recognizing patterns or signatures corresponding to specific physical contexts across a wide array of application verticals. Such sensing becomes even more effective when implemented in a distributed manner across multiple IoT devices in a network, deployed across various vantage points [10].

However, as with many other sensing modalities, sophisticated sensing tasks require complex inference models that are challenging to execute on resource-constrained IoT devices. A common practice is to offload the computation to the network edge. The drawback of such approach is that IoT devices must continuously stream sensory data (e.g., CSI samples) to the network edge while sharing network resources with regular data traffic. This issue is exacerbated in a distributed sensing scenario where multiple sensory data streams coexist. As discussed in Sec. 4, we present a couple of use cases where CSI samples from four Wi-Fi devices are streamed simultaneously to an edge server. Each CSI sample comprises of 64-bit complex values for each of 64 OFDM subcarriers. With an average sampling rate of 800 Hz/device, the aggregate network traffic approaches 20 Mbps. Existing literature employing alternative hardware platforms, such as the Intel 5300 [1] or PicoScenes [3], also reports similar substantial data usage. First, the network footprint taken up by RF sensory data can be substantial and affect the overall QoS of the network. Second, such data has stringent latency requirements in order for the inference models to work accurately. Consequently, as more devices compete for network resources (both data and sensory streams), overall system performance is affected. Third, the continuous pre-processing and transmission of sensory data is energy-intensive, especially for battery-powered IoT devices.

While the aforementioned points suggest potential bottlenecks in IoT-based wireless sensing systems, our paper critically examines these issues. We explore the following question: *To what extent can we improve the network utilization of distributed wireless sensing systems without significantly compromising the inference accuracy of the sensing tasks?* Specifically, we investigate the optimal rate for subsampling the CSI stream and determine the minimum precision (bit-width) required for CSI samples while ensuring that the model performs within acceptable limits.

In this paper, we explore sensory data compression in the light of two sensing use cases: (a) Human Activity Recognition (HAR) within an indoor space of area 1300 sq ft., spanning four activities: walking, running, jumping and unoccupied room, and (b) Road traffic surveillance, primarily counting the number of vehicles passing

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

https://doi.org/10.1145/nnnnnnn.nnnnnnn

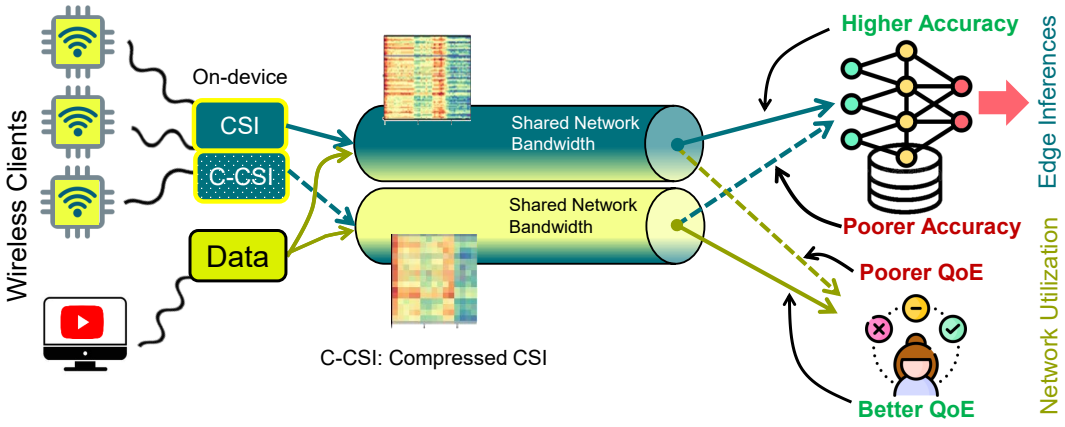


Figure 1: In the above figure, multiple IoT devices send CSI data to the edge server for sensing tasks. However, the same network bandwidth is also shared among regular communication clients. While an increased number of sensors aids in better sensing accuracy, it also impacts the available network bandwidth shared with the communication clients. In this work, we present a comparison of various methods that improve network utility while transmitting CSI data to the edge. We compare subsampling along with various compression methods to provide sufficient link capacity to communication clients while achieving acceptable sensing accuracy. Our goal is to achieve an acceptable compression of CSI data for sensing, enabling both sensing and communication tasks to coexist without poor QoE.

through an intersection. We train relevant deep learning models to be run on an edge device for inference tasks – an LSTM-based network for (a) and a CNN-based network for (b). The compression of CSI samples take place on the device itself.

We implement four compression strategies: uniform quantization, K-Means clustering, Principal Component Analysis(PCA) based approach and Encoder-Decoder based compression. Note, although compression can save network resources, it can be costly in terms of computational resources. For instance, popular deep learning-based compression techniques, such as Encoder-Decoder networks, are resource-intensive and require specific fine-tuning to be effectively utilized on IoT devices. We make the following contributions in this paper.

- We conduct a comparative analysis of four data compression techniques on CSI data (as well as subsampling) and evaluate their effects on network utilization as well as their impact on wireless sensing performance (fig. 4 and fig. 5).
- Our observations indicate that non-uniform quantization is often more effective for simple models compared to uniform quantization, while PCA or *encoder-decoder*-based techniques, although effective, are heavy on resource consumption (fig. 6).
- We present extensive results based on traces collected from real testbed scenarios—one indoors and another outdoors.

2 WIRELESS SENSING

Wireless sensing has gained significant traction from the research community due to its lightweight nature compared to camera-based methods, its ability to utilize existing wireless network infrastructure, and its versatility in various contexts, including non-line-of-sight scenarios – all without the need for additional hardware. In this paper, we focus on Wi-Fi-based sensing using the Channel State Information (CSI) metric. However, similar principles can be applied to other modalities such as Bluetooth, Ultra-Wideband, or millimeter waves.

2.1 Wi-Fi Channel State Information (CSI)

Channel State Information (CSI) is a complex-valued function that encodes how a signal propagates from a transmitter to a receiver. In a MIMO system with N_t transmit antennas and N_r receive antennas, the Channel State Information (CSI) matrix $\mathbf{H}$ consists of elements $\mathbf{h}_{ij}$, each a vector of length K. Here, K represents the number of OFDM subcarriers. The matrix $\mathbf{H}$ matrix can be expressed as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_{11} & \mathbf{h}_{12} & \cdots & \mathbf{h}_{1N_t} \\ \mathbf{h}_{21} & \mathbf{h}_{22} & \cdots & \mathbf{h}_{2N_t} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_{N_r1} & \mathbf{h}_{N_r2} & \cdots & \mathbf{h}_{N_rN_t} \end{bmatrix} \tag{1}$$

where each $\mathbf{h}_{ij}$ is a vector of length K:

$$\mathbf{h}_{ij} = \begin{bmatrix} h_{ij}(1) \\ h_{ij}(2) \\ \vdots \\ h_{ij}(K) \end{bmatrix} \tag{2}$$

Here, $\mathbf{h}_{ij}$ is the complex CSI value for each of the K subcarriers between the j-th transmit antenna and the i-th receive antenna. For our setup, which utilizes Wi-Fi over a 40 MHz bandwidth, k=64, corresponding to sixty four OFDM subcarriers. Also, for our setup both N_t and N_r equal to one, i.e., $\mathbf{H} = \mathbf{h}_{11}$. However, not all of the sixty four subcarriers are utilized for data transmission. Some subcarriers are designated as null subcarriers while others are pilot subcarriers, used for channel estimation and synchronization. A

total of fifty two subcarriers carry the actual data payload which are used as the sensory data.

2.2 Distributed Wireless Sensing

By leveraging the detailed signal properties captured in CSI, it is possible to monitor and analyze the environment for various activities and events without relying on traditional sensors. In a distributed setup, multiple wireless devices throughout the area of interest can collaborate, providing comprehensive coverage and more accurate sensing capabilities. This method allows for more robust detection of movements, human occupancy, activity, all while maintaining user privacy and reducing the need for intrusive hardware. Also distributed sensing can improve the coverage area where a sensing solution is deployed. In such a setup, all devices continuously stream complex CSI values to the edge server. The edge server incorporates appropriate fusion mechanism to aggregate the CSI data and perform inference tasks.

2.3 A Case for Diminishing Returns

In the context of distributed wireless sensing, diminishing returns refer to a point where increasing the precision (bit-width) of CSI values or the sampling rate results in minimal or no significant improvement in the accuracy of inferences. Beyond such point, the sensing system consumes more resources – higher network bandwidth, computational load, and energy usage, however that does not correspond to substantial gains in performance. Instead, such over-precision results in higher resource consumption, including increased network bandwidth, computational load and energy usage. Consequently, while the accuracy of the sensing tasks may remain stable, the overall utility of the system decreases due to the disproportionate rise in resource consumption relative to the benefits gained. Such inefficiency highlights the need to balance data precision, sampling rate, and resource use to keep the sensing system effective and sustainable.

3 CSI COMPRESSION

Consider a distributed wireless sensing setup with N devices recording and continuously streaming Wi-Fi Channel State Information (CSI) data to an edge server for inference. Streaming each individual CSI vector is inefficient and incurs significant network packet header overheads. The incoming CSI data is thus aggregated into batches of m samples, each containing K complex values. These aggregated batches are suitable for compression algorithms, such as quantization, to reduce data size and enhance transmission efficiency by leveraging unnecessary high precision, structure, and historical information. For instance, in `RPi`, both real and imaginary numbers are stored as int16, allowing for $66K$ possible values. However, CSI data typically falls within a smaller range, making it suitable for quantization. The compression techniques utilized in the work are summarized as follows.

3.1 Uniform Quantization

A commonly used uniform quantization function is given by,

$$Q_u(r) = \textsc{Round}\left(\frac{r - Z}{S}\right) \tag{3}$$

where Q_u is the uniform quantization operator, r is the input value, S is the scaling factor and Z is the zero point. Typically, the min-max range of the data, say $[a, b]$, is used to determine the constant $S = \frac{b-a}{2^n-1}$ and $Z = (a + b)/2$, where n is the number of bits required in terms of precision. However, this min-max range is susceptible to outliers in the data. To address this, percentile ranges (e.g., 5^{th} and 95^{th}) are used, with the i^{th} min-max value serving as a, b. On the edge server side, the data is *dequantized* (i.e., original number of bits restored) as $\tilde{r} = S \cdot r + Z$. Note that the dequantized data varies slightly from the original data because of rounding.

3.2 Non Uniform Quantization with K-Means

As shown in fig. 2, generally and in our case too, the obtained CSI data exhibits sparsity and clustering around a few values. However, these clusters are non-uniformly spaced, making uniform quantization less effective. To utilize this structural characteristic of CSI data, we propose non-uniform quantization. In this approach, cluster centers or means $c_1, c_2, \ldots, c_k$ are selected through uniform random selection and each CSI value is assigned to its nearest cluster center, and cluster centers/means are updated after every iteration to the mean of all datapoints in the cluster, using K-Means, which minimizes the loss function:

$$\min_{c_1,c_2,\ldots,c_k} \sum_i \left\|x_i - c_{x_i}\right\|^2 \tag{4}$$

Here, c_{x_i} denotes the cluster mean closest to x_i. Consequently, each data value x_i can be represented using only $\lceil \log_2 k \rceil$ bits. This method offers substantial improvement: for instance, with $k = 16$, each value can be encoded in just 4 bits, compared to 16 bits as typically used in applications like Raspberry Pi.

In our experiment with 50 CSI samples, each containing 128 values, K-Means typically converges within an average of 25 iterations (standard deviation 8). To optimize efficiency and avoid recomputing means for every batch, we store the K-Means loss $L = \sum_i \left\|x_i - c_{x_i}\right\|^2$ from the previous batch. After each iteration, if the new loss L' falls within a threshold α times L we stop the compression process (see algorithm 1). This approach with $\alpha = 1.05$, reduces the average number of iterations to 3.5.

3.3 Principal Component Analysis (PCA)

Not only are the CSI values sparse and clustered on the real line but they also show sparsity in the Euclidean space. This means that a set of these CSI vectors can be represented in a much smaller subspace using only a few components. However, quantization techniques are not able to take advantage of this, as they truncate each value individually and ignore the overall structure. To exploit this internal structure of the data points, we use Principal Component Analysis to compute the principal components of the CSI vectors, and projections of these vectors onto the principal components are only utilized for further sensing tasks at the edge device.

3.4 Encoder-Decoder based Compression

Using an encoder-decoder architecture can significantly enhance data transmission efficiency. By employing an encoder at the transmitting IoT devices, the high-dimensional CSI data is compressed before being sent. This compression reduces the amount of data

Algorithm 1: CSI Batch Compression: Non Uniform Quantisation with K-Means

Input: R, L
/* Batch of CSI values, Loss threshold */
Output: A, $C = [c_1, c_2, \ldots, c_k]$, L_{new}
/* Assignment of R to cluster centers, Cluster centers, Updated loss threshold */
$C \leftarrow$ `initialise_random_cluster_centers`
$A \leftarrow$ `nearest_cluster`(R, C)
do
  $A' \leftarrow A$
  $C \leftarrow$ `compute_centroid`(R, A')
  $A \leftarrow$ `nearest_cluster`(R, C)
  $L' \leftarrow$ `loss`(R, A)
while $(A' \neq A) \wedge (L' \geq \alpha \cdot L)$
if $A' = A$
  $L_{\text{new}} \leftarrow L'$
else
  $L_{\text{new}} \leftarrow L$
return A, C, L_{new}

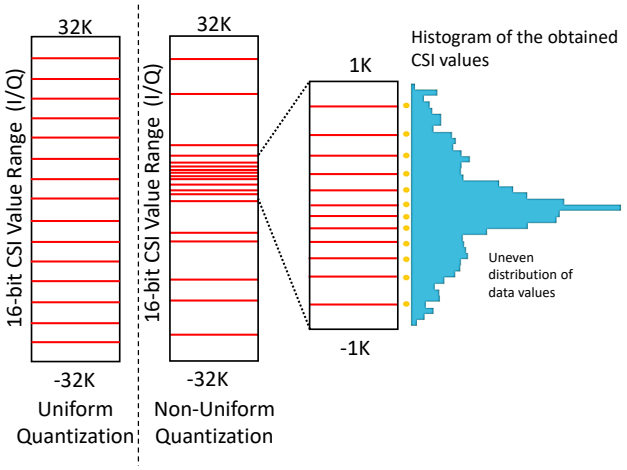


Figure 2: Effect of Quantization. Majority of the values lie in a certain interval. In such cases, uniform quantization will lead to a decreased efficiency in representing the data points.

that needs to be transmitted, conserving bandwidth and reducing latency. Once the compressed data reaches the edge server, a decoder reconstructs the original high-dimensional CSI data. This approach allows for various sensing tasks to be performed accurately at the edge server while minimizing the transmission overhead. The encoder-decoder framework thus facilitates effective CSI compression, enabling robust Wi-Fi sensing even in environments with limited transmission capacities. Our autoencoder consists of two convolutional layers followed by a *maxpool* layer. We use *ReLU* activation function after each layer to introduce non-linearity.

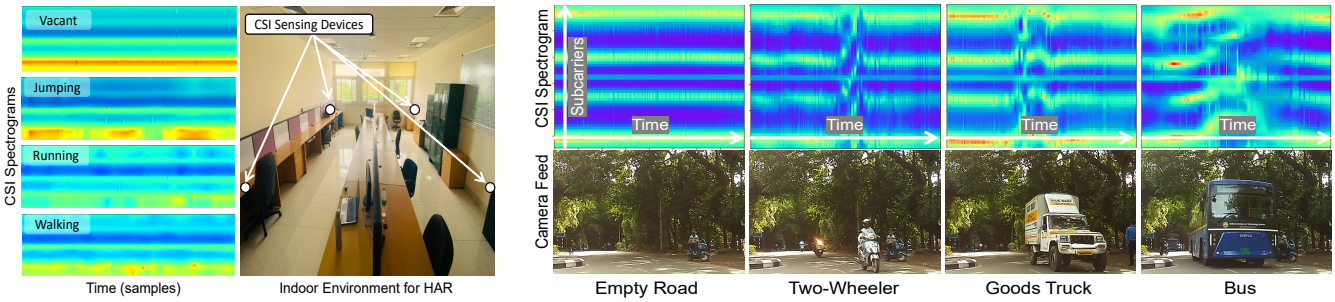
4 TESTBED AND EVALUATION RESULTS

In this section, we discuss the details of our empirical studies for evaluating the quantization schemes. In our setup, we use two types of devices for collecting CSI data, representing two classes of IoT devices: a relatively less provisioned `ESP32-S2` device (`ESP`), and a more computationally provisioned `Raspberry-Pi 4` (`RPi`). To avoid interference the `RPis` and the `ESPs` are tuned to separate channels in the 2.4 GHz band. We use a network of *three* devices of each type to create the distributed sensing setup. The `ESP` provides CSI samples as vectors of 16-bit complex IQ values with a 8-bit in-phase and a 8-bit quadrature component. The `RPi` provides 32-bit samples with 16-bit inphase and a 16-bit quadrature component. While `ESP` provides native support for reading the CSI data directly from the chip, for `RPi` no such native support exists. We use the `Nexmon` toolkit [2] that enables CSI extraction using its modified wireless driver geared towards specific chipsets (Broadcom bcm43455c0). The `ESP` devices can stream CSI data at a maximum rate of 80–100 samples/second, while `RPi` achieves almost an order of magnitude higher rate of 700-800 samples/second. In particular, we deploy two testbeds – one indoors and one outdoors.

Indoor Setup (HAR): We monitor and recognize human activity in a relatively large indoor area of 1300 sq.ft for approximately thirty minutes. We simultaneously collect CSI streams from the `ESP` and `RPi` devices. We infer four different scenarios - walking, running and jumping of a human being along with detecting a human unoccupied room. We train a deep neural network based on Long Short Term Memory or `LSTM` with the CSI samples. We use this trained model to benchmark the HAR performance after applying various compression strategies on the test CSI data.

Outdoor Setup (TRS): In this setting, we monitor our campus road traffic and count the number of vehicles passing an intersection. We place two `nexmon`-enabled `RPi` receivers at a distance of 5 meters on one side of the road, which record the CSI. Midway between them, on the opposite side of the road, we place the wireless transmitter, which also generates Wi-Fi traffic using the `iperf3` tool. Note that `ESP` devices are not used in this experiment due to excessive packet loss, particularly when Non-Line-of-Sight blockages occurred due to passing vehicles. In this setup, we collect approximately 45 minutes of data, including a video stream used for ground truth. During this period, we observe a total of around 1,000 vehicles (including cars, trucks, buses, two-wheelers, bicycles, etc.). We train a Deep Convolutional Neural Network (`DCNN`) on two types of CSI data: one indicating the presence of a vehicle and the other indicating the absence of one.

While training the above models, no quantization was applied to the data. It is also important to note that, in this work, we do not consider compression of the model itself—such as weight quantization or pruning—as discussed in [7]. We achieve an overall testing accuracy of 83% for Human Activity Recognition (`HAR`) and an accuracy of 91% for Traffic Recognition Scenarios (`TRS`). Next, we investigate the effects of various compression schemes on both network load and inference accuracy. Since the `RPi` traffic creates a significantly higher network load compared to the `ESP` devices, we restrict ourselves only to `RPi` traffic for evaluating improvement in network utility. We use the `ESP` device to benchmark resource (memory and energy) consumption of the compression schemes.



(a) Indoor sensing scenario (HAR)

(b) Outdoor sensing scenario (TRS)

Figure 3: Testbed setup for indoor and outdoor scenarios. The CSI data, along with the ground truth video footage, are recorded locally and collated later to perform trace-driven analysis.

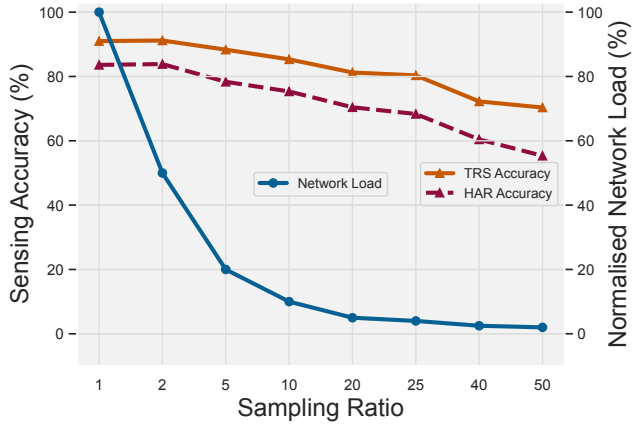


Figure 4: Subsamping the CSI stream effectively brings down the network load, however deteriorates inference accuracy.

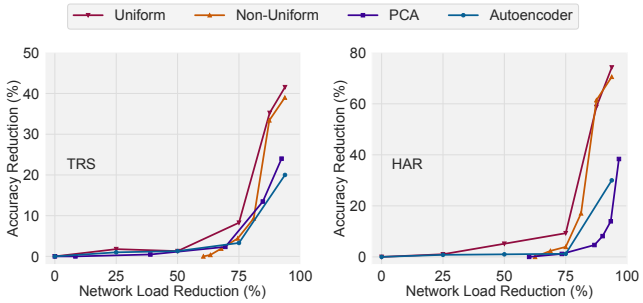


Figure 5: The four CSI compression strategies that we explore in this paper and their effect on network load and inference accuracy.

4.1 Effect of Subsampling

In fig. 4, we illustrate the impact of CSI subsampling on the sensing accuracy of the system as well as the reduction in network load. A *sampling ratio* of x indicates that one in every x CSI samples is sent to the edge for inference tasks. This approach significantly improves network utilization by decreasing the network load, to as much by ≈90%. However, the inference accuracy deteriorates (15–20%) as more temporally sparse CSI samples are streamed to the edge servers. The TRS case, which involves two classes, suffers less in terms of accuracy compared to the HAR case. Subsampling is a relatively straightforward approach to improving network utility. However, more complex sensing tasks may not be able to sustain higher sampling ratios while maintaining accuracy.

4.2 Impact of Data Compression

In fig. 5, we demonstrate the effect of data compression techniques, as discussed previously in Section 3, for both HAR and TRS scenarios. As we apply higher degrees of compression, the network load reduces; however, this also results in a dip in inference accuracy (from an average of 85% to ≈65%). For TRS, the compression strategies have similar performance, except for uniform quantization, which suffered the most. Conversely, for HAR, given the complexity of the models, quantization leads to a drastic drop in accuracy, with the autoencoder-based technique being the least affected.

4.3 Impact on Resource Consumption

Finally, in fig. 6, we illustrate the impact of the compression algorithms on the device's resource consumption. These results correspond to benchmarks done on the ESP device. We use TensorFlow Lite (`tflite` [4]) framework to generate and host the *Encoder-Decoder* model on the ESP platform. Since the compression must occur on the device side, it is essential to ensure that the compression process does not consume equal or more resources compared to not compressing the CSI data.On the left, we present the on-device SRAM that is free after running each of the compression schemes. As we can observe, the *PCA* and *Encoder-Decoder*-based schemes consume the highest amount of memory. In fact, Additionally, the *Encoder-Decoder* leaves very little room in the SRAM (≈60 KB) making it difficult to store further runtime data. Also, *Encoder-Decoder*-based compression results in higher current draw, which translates to increased power consumption compared to the other schemes. Hence, for very complex models and stringent accuracy requirements, *Encoder-Decoder* methods are preferred; however, as we demonstrate, they are not resource-friendly.

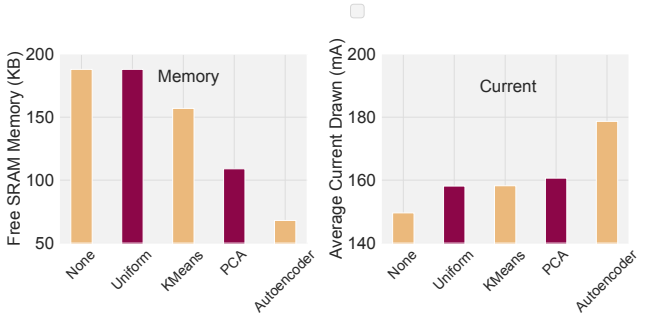


Figure 6: Resource consumption in terms of memory and current-draw for various compression schemes.

5 RELATED WORKS

While a number of works [6, 8, 9, 13] compress the CSI for massive MIMO feedback, only a handful of works [5, 11] explore the possibility of CSI compression for wireless sensing or CSI-based sensing.

The authors in [11] propose `EfficientFi`, a framework designed to compress CSI data, restore it for further use, and recognize activities simultaneously. It employs a quantized feature learning algorithm and a consensus learning framework, distributing tasks between edge devices and cloud servers to optimize performance. The authors evaluate `EfficientFi`'s performance in terms of compression rate, data restoration quality, and recognition accuracy. The authors in [5] introduce `RSCNet`, an architecture that combines CSI compression and sensing, addressing the high dimensionality and transmission overhead of CSI data. `RSCNet` is divided into edge and cloud models, where the edge model compresses CSI windows through an encoder, while the cloud model performs human activity recognition (`HAR`) using a multi-layer perceptron (MLP) classifier. The evaluation criteria include recognition accuracy for `HAR` and Normalized Mean Square Error for compression performance. The works [12, 14] investigate the potential of Wi-Fi based sensing at low transmission rates. In both papers, the authors suggest using Generative Adversarial Networks (GANs) to reconstruct high-rate signal data from low-rate inputs, reducing the need for high packet rates to achieve effective sensing. These works also offer an alternative perspective on compressed sensing, demonstrating that controlling the rate at which CSI is transmitted to the server can still yield acceptable accuracy for sensing tasks. While majority of the techniques employ deep learning for compression, we show that they are quite resource intensive and are not required for simple inference tasks. In this work, we look into these techniques, along with relevant encoder-decoder compression.

CSI Compression for Massive-MIMO: Works [6, 8, 9, 13] propose CSI compression for massive MIMO feedback for variety of networks like 5G and 6G networks. The compression of CSI for massive MIMO lies out of the scope of this work so we will not discuss it here.

6 CONCLUSION

In this work, we explore Edge-assisted inferencing for wireless sensing applications, where CSI data estimated at IoT devices (wireless receivers) are transmitted to Edge devices. As the number of devices (sharing the same wireless medium) scales, this data transmission increasingly consumes network capacity, leading to congestion. We investigate methods to enhance network efficiency by downsampling or quantizing the CSI data and assess the impact on overall inference accuracy. We present with four methods: Uniform Quantization, K-Means Clustering, PCA and Encoder-Decoder based compression for evaluating multiple parameters: accuracy, memory consumption and power required. To our knowledge, this is the first work evaluation non-deep-learning based compression methods for CSI-based sensing. We observe that accuracy is most affected by uniform quantization with drop of more than 40%, so is the observation for K-Means clustering, while the encoder-decoder based compression achieves maximum accuracy even with low amount of CSI-data.

REFERENCES

[1] [n. d.]. Intel® WiFi Link 5300. https://www.intel.com/content/www/us/en/support/products/70971/wireless/legacy-intel-wireless-products/intel-wireless-series/intel-wifi-link-5300.html. Accessed: 2024-07-31.
[2] [n. d.]. nexmon-csi. https://github.com/seemoo-lab/nexmon_csi. Accessed: 2024-07-31.
[3] [n. d.]. PicoScenes. https://ps.zpj.io/. Accessed: 2024-07-31.
[4] [n. d.]. TensorFlow Lite. https://www.tensorflow.org/lite. Accessed: 02-08-2024.
[5] Borna Barahimi, Hakam Singh, Hina Tabassum, Omer Waqar, and Mohammad Omer. 2024. RSCNet: Dynamic CSI Compression for Cloud-based WiFi Sensing. *arXiv preprint arXiv:2402.04888* (2024).
[6] Qiuyu Cai, Chao Dong, and Kai Niu. 2019. Attention model for massive MIMO CSI compression feedback and recovery. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–5.
[7] Manoj Lenka and Ayon Chakraborty. 2024. On-Device Deep Learning for IoT-based Wireless Sensing Applications. In *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 568–574.
[8] Faris B Mismar and Aliye Özge Kaya. 2024. Compression of the Channel State Information with Deep Learning. *arXiv preprint arXiv:2406.14668* (2024).
[9] Muhammad Karam Shehzad, Luca Rose, and Mohamad Assaad. 2021. Dealing with CSI compression to reduce losses and overhead: An artificial intelligence approach. In *2021 IEEE international conference on communications workshops (ICC Workshops)*. IEEE, 1–6.
[10] Xuanzhi Wang, Kai Niu, Jie Xiong, Bochong Qian, Zhiyun Yao, Tairong Lou, and Daqing Zhang. 2022. Placement matters: Understanding the effects of device placement for WiFi sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 1–25.
[11] Jianfei Yang, Xinyan Chen, Han Zou, Dazhuo Wang, Qianwen Xu, and Lihua Xie. 2022. EfficientFi: Toward Large-Scale Lightweight WiFi Sensing via CSI Compression. *IEEE Internet of Things Journal* 9, 15 (2022), 13086–13095. https://doi.org/10.1109/JIOT.2021.3139958
[12] Kun Yang, Xiaolong Zheng, Jie Xiong, Liang Liu, and Huadong Ma. 2022. WiImg: Pushing the Limit of WiFi Sensing with Low Transmission Rates. In *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 1–9. https://doi.org/10.1109/SECON55815.2022.9918170
[13] Qianqian Yang, Mahdi Boloursaz Mashhadi, and Deniz Gündüz. 2019. Deep convolutional compression for massive MIMO CSI feedback. In *2019 IEEE 29th international workshop on machine learning for signal processing (MLSP)*. IEEE, 1–6.
[14] Xiaolong Zheng, Kun Yang, Jie Xiong, Liang Liu, and Huadong Ma. 2024. Pushing the Limits of WiFi Sensing with Low Transmission Rates. *IEEE Transactions on Mobile Computing* (2024).