SpikeCSI: CSI Feedback Compression for MIMO Wireless Systems using Spiking Neural Networks

Eduardo David Lotto*§, Eleonora Cicciarella†§, Francesco Restuccia‡, Francesca Meneghello‡

* Department of Mathematics "Tullio Levi-Civita", University of Padova, Italy
† Department of Information Engineering, University of Padova, Italy
‡ Institute for Intelligent Networked Systems, Northeastern University, United States

***Abstract*—Multi-input, multi-output (MIMO) systems are key for next generation wireless networks to increase capacity without widening the operational bandwidth. The higher the number of antennas, the higher the capacity of the wireless system. However, this requires increasing the amount of control data – the channel state information (CSI) – that the receiver should feed back to the transmitter to properly precode data streams. In this paper, we propose SpikeCSI, a new approach for CSI compression and reconstruction that relies on spiking neural networks (SNNs) to reduce the computational complexity of the procedure. By operating through spikes rather than real numbers, SpikeCSI allows reducing the energy consumption required for MIMO channel sounding. We implement SpikeCSI and compare its performance with state-of-the-art approaches using synthetic data obtained through ray tracing simulations and real data collected through a software-defined radio (SDR)-based mmWave testbed. The results show that SpikeCSI approaches and, in some configurations, even outperform traditional methods while reducing neuron activity in the SNN compared to standard artificial neural networks (ANNs) which rely on dense processing.**

***Index Terms*—MIMO, Channel Sounding, CSI Estimation, CSI Feedback, Spiking Neural Networks.**

I. Introduction

Multi-input, multi-output (MIMO) is key to increasing throughput in wireless networks through *spatial multiplexing*, i.e., simultaneously transmitting multiple data streams in the same time and frequency resources. However, the benefits of MIMO come with the challenge of estimating, compressing and feeding back the channel state information (CSI) from the receivers to the transmitter. CSI is usually encoded into a high-dimensional matrix whose size depends on the number of antennas at the transmitter and receiver and the operational bandwidth. This prevents scaling the number of antennas and bandwidth [1], [2]. Different communication standards propose various approaches for compressing CSI. In the IEEE 802.11 standard, Wi-Fi devices use singular value decomposition (SVD) and Givens rotations to encode the CSI into a set of values called angles [3].

In recent years, machine learning (ML) has been proposed for CSI compression. In particular, artificial neural networks (ANNs) have great potential in compressing CSI without losing its important characteristics, thereby reducing overhead without compromising system performance [4], [5]. However, the execution of ANN requires significant energy budgets [6]. Even if lightweight ANN models can be explored [7], the high rate at which the CSI should be estimated and fed back (10 ms as recommended in [8])) makes ANN-based compression impractical.

§ Eduardo David Lotto and Eleonora Cicciarella are co-first authors.

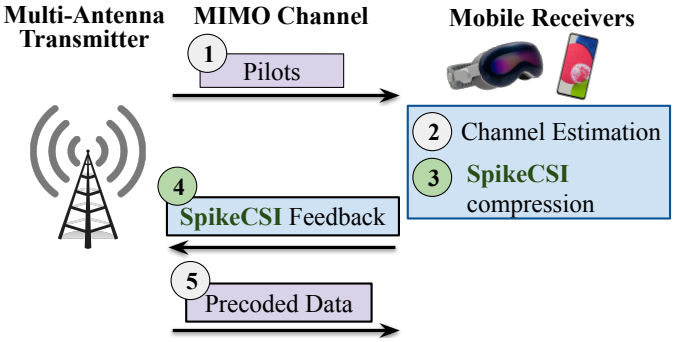

Fig. 1: SpikeCSI MIMO feedback. The numbers indicate the sequence of steps for channel sounding and transmission.

In this paper, we propose SpikeCSI, the first CSI feedback compression mechanism based on spiking neural networks (SNNs). The latter have attracted growing interest due to their event-driven computation, which makes them more energy-efficient than ANNs. As detailed in Section IV-A, while ANNs process dense data structures, SNNs works with spikes, which are binary variables that trigger the execution of an operation only when they take value 1. This allows reducing the number of accumulate (ACC) and multiply-accumulate (MAC) operations required to execute the task and makes SNNs a very promising candidate for MIMO feedback compression. Figure 1 summarizes SpikeCSI. After receiving pilot symbols from the multi-antenna transmitter, the mobile receivers estimate the channel and use the SpikeCSI encoder to compress and feed back the CSI efficiently. The transmitter uses the SpikeCSI decoder to reconstruct the CSI and uses it to compute the weights for data precoding.

Summary of Novel Contributions:

- We propose SpikeCSI, the first SNN-based encode-decoder architecture for sounding feedback compression in low-power MIMO networks. Our approach relies on sparse operations on spikes rather than dense processing of real numbers;

• We extensively evaluate the communication performance of SpikeCSI-empowered MIMO systems through simulations using Sionna, and emulations using experimental data collected on a software-defined radio (SDR)-based mmWave testbed. Our results show that SpikeCSI performance approaches state-of-the-art methods including the IEEE 802.11 technique used by Wi-Fi devices [3] and a recent learning-based strategy for feedback compression [4];

• We study the spike activity and computational cost of SpikeCSI for feedback compression and reconstruction. Our results show that SpikeCSI performs significantly fewer MAC operations than comparable ANN-based strategies, and on average only about 20% of neurons are active at a given timestep. This directly translates into reduced energy consumption when using neuromorphic hardware.

II. Related Work

One of the first work on CSI compression is CsiNet [9], where the authors use a convolutional neural network (CNN)-based autoencoder to extract a latent representation from the CSI for proper reconstruction at the transmitter. In [10], an online learning framework leverages side information already present in 802.11 to train autoencoders and reduce feedback overhead while preserving compatibility with existing devices [10]. DeepMux focuses on IEEE 802.11ax multi-user multi-input multi-output (MU-MIMO) and orthogonal frequency-division multiple access (OFDMA): the station (STA) feeds back quantized CSI on a subset of subcarriers, and the access point (AP) reconstructs the full CSI with a deep neural network (DNN), jointly optimizing resource allocation [11]. SplitBeam adopts a split-DNN with a bottleneck to shrink the feedback and lighten processing at the STA [4]. Other works refine quantization and training regimes, from dynamic bit allocation that balances accuracy and overhead [12] to data-efficient training with extrapolation and few-shot strategies [13]. In addition, the CSI reporting rate can be adapted to the channel dynamics to reduce the overhead. The intuition is that in quasi-static conditions, an update of the feedback may be unnecessary and it can be deferred to save spectrum and computation. For example, MUTE [14] requests updates only from devices whose channel has changed significantly, while SHRINK [1] makes each device decide whether to transmit the feedback. Notice that feedback compression and sounding rate adaptation are not mutually exclusive approaches.

In this paper, we propose for the first time to use SNNs for CSI compression and reconstruction. We believe that this first investigation opens a new research avenue in low-complex learning-based feedback compression mechanisms. This will finally enable their practical integration into battery-constrained devices.

III. SpikeCSI System Model

We consider a single-user MIMO system entailing an N_{TX}-antenna transmitter and an N_{RX}-antenna receiver. The communication strategy is summarized in Figure 1 and consists of two phases: (i) channel sounding, and (ii) actual data transmission, as described next [15].

Channel sounding is required by the transmitter to properly precode the data streams to be simultaneously transmitted to the user. Note that this number is constrained by the minimum between N_{TX} and N_{RX}. Transmitting more data streams is impossible because it would make the system of equations underdetermined and, in turn, the data streams not separable. We consider an explicit channel feedback mechanism for sounding. The downlink channel is estimated at the receiver device and feed back to the transmitter. This operation is triggered by the transmitter which sends some pilot symbols to the receiver (step 1 in Figure 1). By comparing the received pilot symbols with their known sequence of bits, the receiver is able to estimate the CSI of the downlink link (step 2). The CSI is obtained for each pair of transmitter (TX) and receiver (RX) antennas over all the orthogonal frequency-division multiplexing (OFDM) subcarriers. For subcarrier $k \in \{1, \ldots, K\}$ antennas $i_{\mathrm{TX}} \in \{1, \ldots, N_{\mathrm{TX}}\}$ and $i_{\mathrm{RX}} \in \{1, \ldots, N_{\mathrm{RX}}\}$, the baseband CSI writes as

$$H_{i_{\mathrm{RX}}, i_{\mathrm{TX}}, k} = \sum_{p=0}^{P-1} a_p e^{-j\pi[2k\Delta f \tau_p + i_{\mathrm{RX}} \sin\phi_p + i_{\mathrm{TX}} \sin\psi_p]}, \quad (1)$$

where Δf is the OFDM subcarrier spacing and P is the number of multipath components, and identifies the number of multiple copies of the transmitted signal that are collected at the receiver due to reflections, diffractions, and scattering [15]. The parameters a_p, τ_p, ϕ_p and ψ_p indicate the amplitude, time, angle of arrival (AoA) and angle of departure (AoD) characterizing the p-th multipath component.

The CSI matrix is compressed through our SpikeCSI algorithm (step 3 in Figure 1, see Section IV-B) and fed back to the transmitter (step 4). Once the feedback has been received, the transmitter decodes and uses the CSI to obtain the precoding weights. The latter define the weights to be applied to the different streams at the different transmitter antennas for simultaneous transmission (step 5 in Figure 1). An effective precoding minimizes the bit error rate (BER) at the receiver by properly orthogonalizing the streams and, in turn, minimizing inter-stream interference.

IV. SpikeCSI Channel Feedback Compression

The main objective of our SpikeCSI algorithm is to reduce the burden of the compression at the mobile receivers. This objective is primarily reached through the use of a SNN-based encoder which allows greatly increasing computation efficiency with respect to ANN-based compressors. To further reduce complexity and ease integration into battery-constrained devices, we focus on small (shallow) architectures for the encoder. In this view, we performed a preliminary hyperparameter search to reduce the number of layers used in the encoder SNN.

In the following, we first briefly introduce the SNN neuron's model we adopt in Section IV-A. The description of our SpikeCSI architecture is then described in Section IV-B.

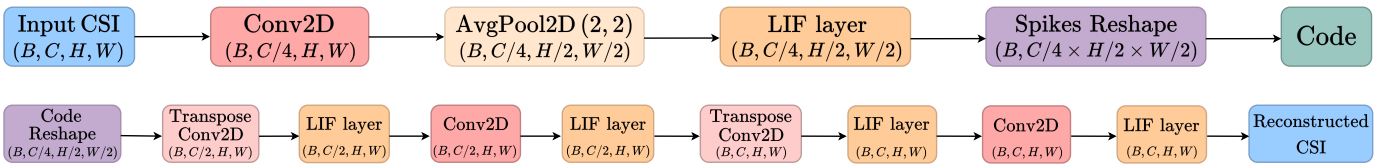

Fig. 2: From top to bottom: SpikeCSI encoder and decoder for compression at the device and reconstruction at the transmitter.

A. LIF-based Spiking Neural Networks

SNNs can be implemented using a variety of neuron models that differ in biological realism and computational complexity. In this work, we adopt the Leaky Integrate-and-Fire (LIF) neuron model [16], which is among the most widely used due to its simplicity and computational efficiency. The LIF neuron describes neuronal activity through a *membrane potential* that integrates synaptic inputs over time while exhibiting exponential leakage. When the membrane potential exceeds a predefined threshold θ, the neuron emits a unitary *spike*, and the membrane potential is instantaneously reset to a resting value. In SNNs, LIF neurons are typically organized into layers. Let $U_i^{\ell}[t]$ denote the membrane potential of neuron i in layer ℓ at discrete time t, $I_i^{\ell}[t]$ its input current, and $S_i^{\ell}[t]$ the corresponding output spike. The discrete-time dynamics of the LIF neuron are formulated as

$$U_i^{(\ell)}[t] = \big(1 - S_i^{(\ell)}[t-1]\big) \cdot \big(\beta\, U_i^{(\ell)}[t-1] + I_i^{(\ell)}[t]\big), \quad (2)$$

$$I_i^{(\ell)}[t] = \mathcal{F}\big(S^{(\ell-1)}[t]\big), \quad (3)$$

$$S_i^{(\ell)}[t] = \Theta(U_i^{(\ell)}[t] - \theta). \quad (4)$$

The term $(1 - S_i^{(\ell)}[t-1])$ in Eq. (2) implements a *reset-to-zero* mechanism: whenever the neuron fires a spike at the previous timestep (i.e., $S_i^{(\ell)}[t-1] = 1$) its membrane potential is set to zero. The parameter $\beta \in (0, 1)$, instead, represents the decay rate of the membrane potential in the absence of input stimuli. The input current $I_i^{\ell}[t]$ can be any transformation $\mathcal{F}$ of the output spikes of the previous layer, including linear combinations, convolutions, or other functions. Finally, Eq. (4) defines the spike generation mechanism, where $\Theta(\cdot)$ denotes the Heaviside step function. Due to the non-differentiability of such function, SNNs cannot be directly optimized using gradient-based methods. To address this, we employ the *surrogate gradient* approach [17], in which the derivative of the Heaviside function is replaced by a differentiable surrogate function during the backward pass, enabling efficient training via gradient descent.

B. SpikeCSI Learning Architecture

The proposed SpikeCSI consists of a convolutional spiking encoder-decoder architecture, as illustrated in Figure 2. The input CSI tensor, originally shaped as $(N_{\text{RX}}, N_{\text{TX}}, K, 2)$, is rearranged by merging the antenna dimensions and treating the subcarriers as the channel dimension. Including the batch size B, the resulting shape becomes

$$(B, C, H, W) = \big(\text{batch_size},\ K,\ N_{\text{RX}} \cdot N_{\text{TX}},\ 2\big).$$

The SpikeCSI encoder module, purposely minimized, reduces the input dimensionality through a single spiking convolutional block. The CSI tensor is first processed by a 3×3 convolutional layer with unit stride and zero padding, reducing the number of channels by a factor of four. Then, an average pooling operation with a 2×2 kernel halves the spatial resolution along both H and W. The resulting features maps are passed through a LIF layer to produce spike-based activations. These spiking features are finally flattened to produce the latent representation of size $C/4 \times H/2 \times W/2$, forming the compressed code. Therefore, the size of the code is not fixed, as it adapts to the input dimensionality.

For example, in a 8×4 MIMO system with 1024 subcarriers (80 MHz bandwidth), the latent size becomes $1024/4 \times (8 \cdot 4)/2 \times 2/2 = 4096$. Since the code consists exclusively of binary spikes, its transmission requires 4096 bits, with each spike encoded using a single bit. This represents a considerable reduction in transmission overhead compared to the IEEE 802.11 compression scheme [3], which would require more than twice the number of bits (8448 bits) for the same configuration with the highest quantization.

The SpikeCSI decoder reconstructs the input by progressively upsampling and refining the compressed representation through a sequence of transposed and standard convolutional layers, each followed by a LIF layer to model the spiking dynamics. The first transposed convolution has $C/2$ output channels and employs a 2×2 kernel with stride 2, thus doubling the spatial resolution. This is followed by a 3×3 convolution that refines the feature maps while preserving their dimensions. A second transposed convolution with unit kernel and unit stride restores the number of channels to C. Finally, a 3×3 convolution produces the output feature maps, which are processed by the last LIF layer.

To exploit spiking dynamics, the network is trained for $T = 2$ timesteps. At each timestep, the LIF neurons update their membrane potentials based on the incoming input and their internal states, and generate output spikes according to the thresholding mechanism. Each LIF neuron has learnable firing threshold θ and membrane potential decay rate β, with initial values set to 0.2 and 0.9, respectively. After spike emission, all neurons reset their membrane potential to zero, except those in the final LIF layer. Indeed, for the latter, the reset mechanism is disabled, allowing membrane potentials to accumulate over timestep; the final reconstruction is then obtained from these accumulated membrane potential at the last timestep.

During training, the gradients are approximated using the fast sigmoid function $f(x) = x/(1 + |x|)$, and the network is

optimized using the Adam algorithm with a learning rate of 10^{-3}. Finally, the loss function is defined as the mean squared error (MSE) between the membrane potential of the last LIF layer at $T = 2$ and the original CSI tensor.

V. Data Collection and Evaluation Setups

We first train and evaluate the system in a simulated environment using synthetic data. Hence, we test the SpikeCSI strategy also on real data collected using a fully digital SDR-based testbed. Additional details are provided in the following. We will release the entire datasets and code for reproducibility.

A. Synthetic Dataset using Sionna

We use the Sionna ray tracer [18] to generate controlled datasets to train our feedback compression algorithm. The indoor space utilized for wireless communication simulations is developed in Blender with the intention of creating a realistic and physically accurate 3D scene compatible with Sionna. The virtual room measures 6×8 m^2 with a ceiling height of 2.8 m. The structural shell consists of fully enclosed walls, a floor, and a ceiling. To enhance realism and introduce potential sources of reflection, several pieces of furniture and interior elements are added: two wooden tables, three wooden chairs, two metallic shelves, a door, and windows. Additionally, architectural details such as skirting and plasterboard coving are included to more accurately represent a typical indoor environment. The material mapping process within Blender is one of the most significant steps in the modeling procedure. For each object in the scene, a name for the material is assigned so that, when the environment is brought into Sionna, the materials are automatically assigned to Sionna's pre-existing `RadioMaterials`. These inherit all the electromagnetic properties necessary to simulate the interaction of radio waves with objects composed of specific materials. For example, wood and metal reflect and absorb in different ways, and accurate material allocation ensures that their impacts are well represented when the channel is modeled. This is needed to produce realistic CSIs during the dataset creation. To integrate the modeled environment into Sionna, the Blender scene is exported in Mitsuba XML format. This format preserves the mesh geometry, spatial transformations, and material mapping information, ensuring that both the geometric structure and the electromagnetic properties of each object are retained. This scene is subsequently used as the base environment for generating multiple propagation scenarios by varying the positions and configurations of transmitters and receivers, thus enabling the creation of diverse multipath conditions.

1) Communication Link: We simulate a downlink communication scenario at a carrier frequency of 5.32 GHz. We consider a single-user MIMO link where both the TX and the RX are equipped with `PlanarArray` antennas following the Third Generation Partnership Project (3GPP) TR 38.901 standard pattern, with single polarization [19]. The transmit antennas is configured with $N_{\mathrm{TX}} \in \{4, 8\}$ while the number of receive antennas is selected in the set $N_{\mathrm{RX}} \in \{2, 4\}$. This allows assessing the effect of spatial diversity on channel estimation. The antenna spacings is set to half the wavelength.

We consider 80 MHz of bandwidth. The system operates through OFDM with a subcarrier spacing of 78.125 kHz and OFDM subcarriers $K = 1024$. A total of 14 OFDM symbols per slot are transmitted using Quadrature Phase Shift Keying (QPSK) modulation (2 bits per symbol). The allocation of subcarriers and OFDM symbols is defined through a `ResourceGrid` which specifies the mapping of data and pilot symbols in the time–frequency domain. These pilots, placed according to a Kronecker pattern and corresponding to the 2^{nd} and 11^{th} OFDM symbols, are used for channel sounding so that the Least Squares estimator computes the channel coefficients in the frequency domain, providing the estimated CSIs. The number of transmitted spatial streams is set equal to N_{RX} to ensure full MIMO utilization.

These channel data obtained from the two pilot subcarriers are split into two separate tensors, each corresponding to a different time step. Consequently, each simulated TX–RX configuration produced two distinct dataset entries, increasing the dataset size while preserving temporal diversity. However, since both entries originate from the same underlying channel realization, care must be taken during dataset partitioning to avoid data leakage. Specifically, both samples derived from the same TX–RX position must be placed in the same split (training, validation, or test), ensuring that the model does not encounter the same channel condition in both training and evaluation phases.

To produce a diverse set of channel realizations, both TX and RX are randomly moved across a predefined three-dimensional grid of positions within the simulated room. The grid covers the following range of spatial coordinates along the X, Y, and Z axes:

- X: from -2.7 m to 2.8 m with a step of 0.5 m;
- Y: from -3.8 m to 3.8 m with a step of 0.76 m;
- Z: from 0.3 m to 2.4 m with a step of 0.7 m.

For each simulation run, the TX and RX positions are randomly shuffled to avoid spatial bias in the dataset. Certain positions are excluded to ensure valid line-of-sight (LOS) and to avoid unrealistic placements (e.g., inside furniture or outside the intended simulation area). Both the TX and RX are assigned small random velocities to emulate motion effects. The `PathSolver` Sionna function is used to compute the propagation paths between the TX and the RX, considering the LOS and first-order specular reflections. For each valid TX-RX pair, the number of resolved propagation paths is computed. During preliminary experiments, we observe that most samples correspond to configurations with 4–7 paths, while scenarios with fewer or more paths are under-represented. To avoid this imbalance, the dataset construction process enforces a target distribution across path-count classes, ensuring that each class contributes the same number of samples. This prevents the models from being trained predominantly on a narrow subset of propagation conditions, while encouraging them to generalize better across varying multipath richness.

Figure 3 shows an example of TX and RX placements within the simulated indoor environment. The red sphere represents the TX position, the green sphere represents the RX position, and the white lines correspond to the resolved propagation paths between them as computed by the ray-tracing simulation.

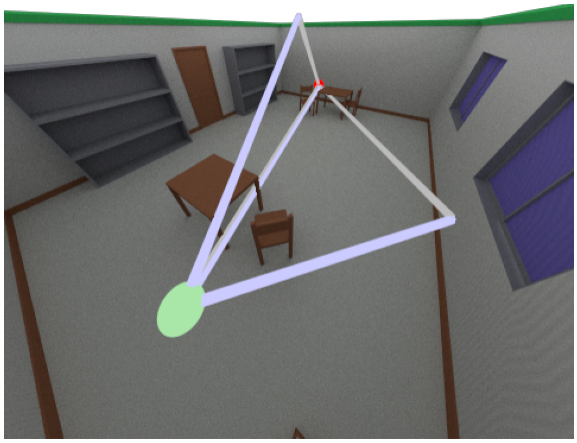

Fig. 3: Example of Sionna ray tracing in the designed room.

2) Data Description and Preprocessing: The data collected for each pair of positions and for each pilot time step includes the multipath components from the ray tracer, and the CSI estimated from the transmission system. Both the multipath components and the CSI are complex-valued data, which must be properly handled before being processed by a neural network. To this end, we split each complex number into its real and imaginary components.

The CSI consists entirely of complex values. Its shape is given by $(N_{\mathrm{RX}}, N_{\mathrm{TX}}, K)$. After splitting the real and imaginary parts, the data is reshaped into $(N_{\mathrm{RX}}, N_{\mathrm{TX}}, K, 2)$, where the last dimension corresponds to the two separate components (real and imaginary) of each complex value. Any sample containing `NaN` values is discarded to preserve data integrity and to avoid introducing artifacts. These values are generated during the Sionna simulation, typically resulting from invalid channel coefficient computations for certain subcarriers or paths under rare geometrical configurations.

Next, we apply a min-max normalization to scale the CSI into a common range. Given a feature value x, its normalized counterpart x' is obtained as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (b - a) + a, \tag{5}$$

where $[a, b]$ is the target normalization range, and $x_{\min}, x_{\max}$ are the minimum and maximum values. The normalization is performed separately for each RX–TX antenna pair, with target range $[-1, 1]$.

3) Overview of the Synthetic Datasets: To evaluate the impact of different channel conditions and system configurations, several dataset variants are generated from the ray-tracing simulations. Table I summarizes the dataset configurations, all of which are balanced with respect to the distribution of path-count classes to ensure fair and unbiased model training. This balancing prevents the models from overfitting to the most common propagation scenarios and improves generalization across a wider range of channel conditions. In our datasets, we set the maximum number of paths to 9 as a trade-off between the desired dataset size and the distribution of simulated samples. Indeed, samples with more than 9 paths were significantly less frequent in the simulations compared to other cases.

TABLE I: Summary of the generated and used datasets with their main characteristics.

Identifier	No. Samples	BW	MIMO	No. Subcarrier K
4×2_80	10,792	80	4×2	1024
4×4_80	9,614	80	4×4	1024
8×4_80	10,776	80	8×4	1024

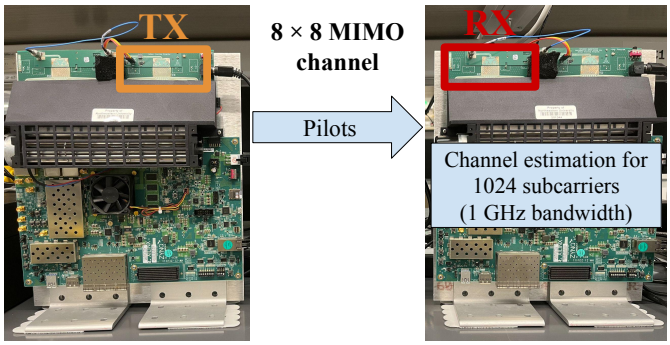

Fig. 4: SDR-based testbed for experimental data collection.

B. Experimental Dataset using the SDR-based Testbed

We collected experimental CSI samples using a fully-digital mmWave SDR built from two Zynq UltraScale+ SoCs with Pi-Radio front-ends [20], as depicted in Figure 4. The system consists of a transmitter and a receiver with 8 radio frequency (RF) chains each, enabling 8×8 multiplexing. A total of 1 GHz of bandwidth is available for transmission in the 57–64 GHz band and allows the creation of OFDM transmissions with 1024 subcarriers. We collect CSI data for different placements of the TX and RX devices. The data is used to emulate the system through Sionna: the collected CSI samples are used as the real channel instead of the ray tracer data. Also in this case, we consider 14 OFDM symbols per slot modulated through QPSK modulation (2 bits per symbol). Pilot symbols are used in the 2^{nd} and 11^{th} OFDM symbols.

VI. Experimental Results

The SpikeCSI framework is implemented in PyTorch, using the snnTorch library [21] to implement SNNs. We evaluate the performance of the SpikeCSI encoder–decoder by analyzing the CSI reconstruction accuracy in Section VI-A. Hence, we implement the channel sounding procedure, MIMO precoding, and data transmission using the Sionna's functions. We consider the entire communication system consisting of the transmitter and the receiver and evaluate the BER at the receiver device after data precoding at the transmitter. The results

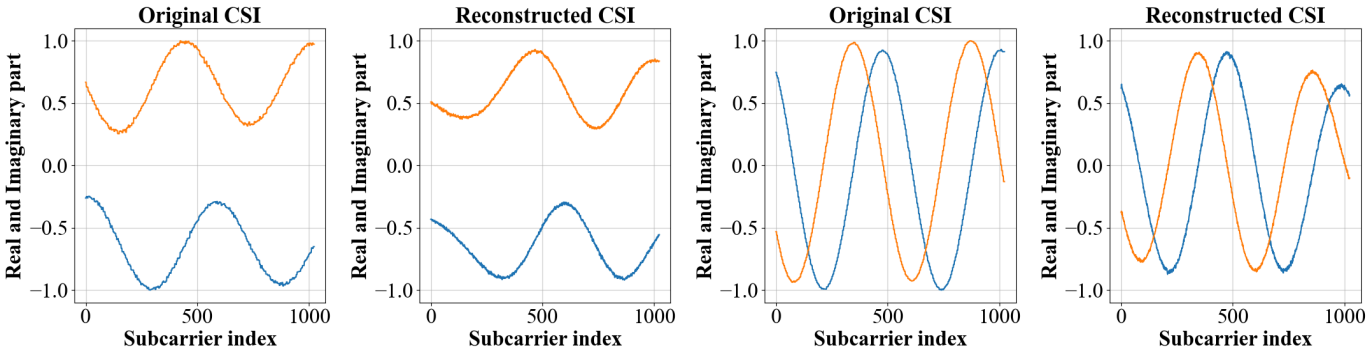

Fig. 5: Example of CSI reconstruction using SpikeCSI on the *4×2_80* test set.

are reported in Section VI-B and Section VI-C for the ray tracer data and the real channel measurements, respectively. We consider four baselines for performance assessment which differ based on the information used to obtain the precoding weights: (i) perfect CSI from ray tracing or data, (ii) CSI estimated on pilots without compression, (iii) Wi-Fi compression based on the IEEE 802.11 standard, which consists of SVD and Givens rotations [3], and (iv) SplitBeam, which uses an ANN-based encoder–decoder architecture for CSI compression and reconstruction [4]. Finally, in Section VI-D we analyze SpikeCSI computational complexity in terms of spike activity and number of MAC and ACC operations.

A. SpikeCSI CSI Reconstruction Accuracy

Table II shows the training and validation losses obtained applying the SpikeCSI compression and reconstruction method on each dataset. The code sizes are also reported in the table. On the *4×2_80* and *4×4_80* datasets, with code sizes 1024 and 2048, respectively, the model achieves very low losses, with validation errors of 0.0207 and 0.0244, respectively. Instead, for the *8×4_80* dataset, where samples consist of $K = 1024$ subcarriers, and the SNN-generated code is of size 4096, the validation error is 0.0287, yet in line with the previous results.

Figure 5 presents two representative reconstructions obtained by the SpikeCSI encoder–decoder on the *4×2_80* test set, on which a MSE of 0.0218 is achieved. In both cases, the reconstructed CSI closely follows the trend of the original one, effectively capturing both the amplitude and phase variations across the subcarriers. This demonstrates the ability of SpikeCSI to learn a meaningful latent representation capable of retaining the essential spectral characteristics of the channel, albeit with a minimal loss of spatial resolution. However, it can also be observed that the reconstructions tend to smooth out some of the finer details, especially in scenarios with more variations or higher frequency oscillations.

B. Bit Error Rate Communication Performance

To evaluate the effectiveness of the SpikeCSI compression and reconstruction technique for precoding, we present a BER analysis comparing the communication system performance using precoding computed from (i) perfect CSI via ray tracing,

TABLE II: SpikeCSI reconstruction loss on the training and validation datasets.

Dataset	Code size	Training loss	Validation loss
4×2_80	1024	0.0122	0.0207
4×4_80	2048	0.0118	0.0244
8×4_80	4096	0.0143	0.0287

(ii) estimated CSI, and (iii) reconstructed CSI obtained with SpikeCSI, alongside precoding computed with the Wi-Fi IEEE 802.11 standard and the SplitBeam technique. The experimental framework is developed using Sionna, and the analysis is carried out on all samples of the datasets reported in Table I, with reconstructed CSI obtained through cross-validation to ensure no data leakage occurs between training and test sets, thereby providing a fair evaluation of the generalization ability of our SNNs at the device and the transmitter.

The information is compressed and reconstructed using SpikeCSI and the other baselines, and the reconstructed version is used to obtain precoding weights. Hence, a set of information bits is encoded, mapped to QPSK symbols, precoded with the computed weights, and transmitted over the channel. At the receiver side, the signal is equalized, demapped, and decoded to recover the bitstream, which is then compared with the transmitted bits to compute the BER. This simulation is repeated for different signal-to-noise ratio (SNR) values and the results are reported in Figure 6, where the BER values are obtained by averaging across all positions. In the 4×2 setup, SpikeCSI performs competitively with the baselines. While it exhibits higher variance at low SNR (0–5 dB) compared to Wi-Fi, it quickly converges to the baseline performance as the SNR improves, matching the reliability of traditional methods at 15 dB. The 4×4 configuration proves to be the most demanding scenario for all evaluated methods. This is primarily because the system operates with four concurrent spatial streams, utilizing the maximum rank of the channel. The resulting high inter-stream interference creates a significant performance bottleneck. Also in this case, SpikeCSI demonstrates remarkable stability. Despite the inherent difficulty of the full-rank 4×4 environment, SpikeCSI maintains a performance envelope that is on par with, and at

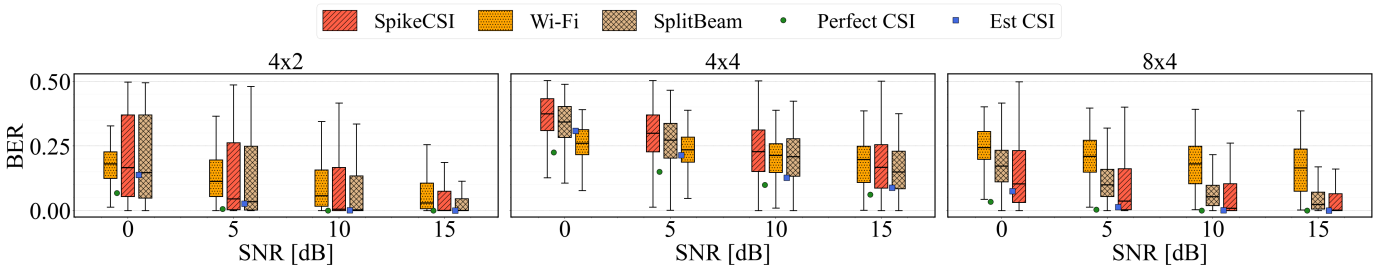

Fig. 6: BER for different single-user configurations evaluated on the simulated scenarios.

high SNR slightly more consistent than, the SplitBeam and Wi-Fi baselines. The true advantage of SpikeCSI emerges in higher-dimensional configurations, such as the 8×4 scenario. In this setting, SpikeCSI significantly outperforms all other methods across the entire SNR range, even at 0 dB. As the SNR increases, SpikeCSI's error rate drops toward zero, effectively converging with the Perfect CSI lower bound. This indicates that SpikeCSI is well-suited for high-dimensional MIMO systems, where it can effectively exploit spatial diversity to suppress noise and interference more efficiently than traditional feedback mechanisms.

C. Experimental Results

Table III summarizes the experimental results obtained by emulating a 4×2 single-user MIMO system using the experimental data collected through the SDR-based setup. We compare the BER achieved performing precoding using SpikeCSI compression strategy with the performance of SplitBeam [4] and the use of the uncompressed estimated CSI. The results are obtained as the average over 17 channel measurements collected in different conditions. The BER values show that SpikeCSI performance approaches SplitBeam, while being 2 orders of magnitude worse than the precoding with the uncompressed CSI. This is due to two main reasons. First, the results are obtained without any channel coding mechanism due to Sionna implementation constraints. Second, we used the SNNs trained on the simulated data to compress and decompress the information at the device and transmitter, respectively. This strategy is desirable for designing plug-and-play solutions that are trained offline on extensive datasets and can be directly deployed on real devices. However, obtaining a solution that generalizes without any adaptation is a very challenging task. Indeed, the statistical distributions of the real channels likely depart from the statistics of the CSI samples used in the offline training. This effect becomes even worse when the number of transmitted streams increases in the 4×4 and 8×4 configurations. The ideal approach to address this issue would be to collect extensive datasets in the real deployment scenarios and use them to train the learning algorithm. In this way, the latter adapts to the actual channel conditions and generates an effective compressing/decompressing pipeline.

D. Computational Complexity

We use the spike activity rate (SAR) as a proxy for energy consumption, since neuromorphic hardware operate in an event-driven manner where computation is triggered only by spike events [6]. As a consequence, a lower SAR directly translates into lower power consumption. The per-layer SAR is defined as the fraction of active (i.e., firing) neurons at each timestep, averaged over time and samples. Across all dataset variants, the highest SAR is consistently observed in the first and last LIF layers, corresponding to the encoding and reconstruction stages, respectively (see Figure 7). This behavior indicates that higher neuronal activity is required to transform the input CSI into a spike-based latent representation and to reconstruct it back. Furthermore, increasing the number of antennas from 2 to 4 and 8 does not lead to a rise in SAR, suggesting that the proposed architecture scales efficiently with input dimensionality. Notably, when averaging the SAR across all layers of the network, the overall activity remains around 0.2 for all datasets. This means that, on average, only about 20% of the neurons emit spikes at a given timestep, while the remaining 80% remain inactive, resulting in sparse spike-driven computation.

TABLE III: BER results for different configurations using the mmWave testbed.

Method	Uncompressed CSI	SplitBeam [4]	SpikeCSI (ours)
BER 4×2	3.5×10^{-4}	4.8522×10^{-2}	7.2769×10^{-2}

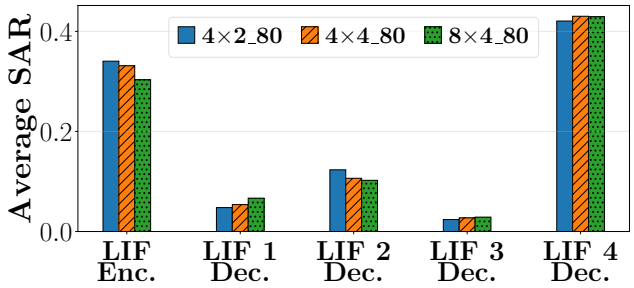

Fig. 7: Per-layer spike activity rate across different datasets.

In addition to the SAR, we evaluate the computational complexity of the network by counting the number of MAC and ACC operations [22]. While the SAR serves as a proxy for energy consumption on neuromorphic hardware, the MAC/ACC counts enable a direct comparison with standard ANNs.

For a 4×2 MIMO system with 1024 subcarriers (80 MHz bandwidth), SpikeCSI performs approximately 38M MACs and 0.1M ACCs per timestep, resulting in a total of 76M MACs and 0.2M ACCs over two timesteps. Despite this temporal processing, the overall computational cost remains significantly lower than that of SplitBeam, which requires around 114M MACs and 0.1M ACCs for the same configuration when using a compression rate of $\kappa = 1/8$, corresponding to a latent dimension of 1024.

Even when the compression rate of SplitBeam is increased, reducing the latent size to 256, its computational complexity remains comparable ($\approx$ 70M MACs). However, this comes at the expense of much stronger compression, whereas SpikeCSI maintains a substantially larger latent size (1024) without increasing the computational cost. In addition, the SNN could potentially benefit from a preliminary spike-based encoding of the CSI, which may further exploit the sparsity of spiking representations. This is an interesting future development of the current investigation.

VII. Concluding Remarks

In this paper, we present SpikeCSI, the first CSI compression and reconstruction mechanism based on SNNs. SpikeCSI-based MIMO precoding performance approaches the one obtained with state-of-art methods while being much less computationally expensive. This is due to the spike-based processing at the core of SpikeCSI. In our proposed SNNs the neurons are active about 20% of the time on average, which is one-fifth of neuron activity in ANNs-based strategies (dense processing, neurons are active 100% of the processing time). We believe that our first investigation opens the door for a broader adoption of SNNs instead of ANNs in wireless devices that are battery-constrained and, in turn, call for low-complex solutions. Future research avenues include a deeper evaluation of the system with real channel measurements and the development of an online adaptation framework for practical system deployment.

Acknowledgments

This work has been supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP) CUP C93C24004880002, Investment 1.2, project "CAMELIA"; by the National Science Foundation under grants CCF-2218845, ECCS-2229472 and ECCS-2329013; by the Air Force Office of Scientific Research under grant FA9550-23-1-0261; and by the Office of Naval Research under grant N00014-23-1-2221.

References

[1] K. Rumman, F. Meneghello, K. F. Haque, F. Gringoli, and F. Restuccia, "SHRINK: Reducing MIMO Feedback Overhead in Wi-Fi with Dynamic Data-Driven Channel Sounding," in *Proc. of ACM MobiHoc*, 2025.

[2] K. F. Haque, F. Meneghello, J. Ashdown, and F. Restuccia, "BANDWEAVE: Enhanced Channel Estimation in MIMO Networks with Multi-Band Fusion," in *Proc. of IEEE INFOCOM*, 2026.

[3] IEEE, "IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN," *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, 2021.

[4] N. Bahadori, Y. Matsubara, M. Levorato, and F. Restuccia, "SplitBeam: Effective and Efficient Beamforming in Wi-Fi Networks Through Split Computing," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 864–874, IEEE, 2023.

[5] F. Qi, J. Guo, Y. Cui, X. Li, C. Wen, and S. Jin, "Deep learning-based CSI feedback in Wi-Fi systems," in *2024 5th International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, pp. 43–48, IEEE, 2024.

[6] F. Akopyan *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[7] R. Mishra and H. Gupta, "Transforming large-size to lightweight deep neural networks for IoT applications," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–35, 2023.

[8] M. S. Gast, *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond*. O'Reilly Media, Inc., 2013.

[9] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive mimo csi feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.

[10] P. K. Sangdeh, H. Pirayesh, A. Mobiny, and H. Zeng, "LB-SciFi: Online learning-based channel feedback for MU-MIMO in wireless LANs," in *Proc. of IEEE 28th International Conference on Network Protocols (ICNP)*, 2020.

[11] P. K. Sangdeh and H. Zeng, "DeepMux: Deep-learning-based Channel Sounding and Resource Allocation for IEEE 802.11 ax," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2333–2346, 2021.

[12] Z. Zuo, Y. Zhong, Y. Li, J. Zhang, and X. Ge, "AIML-Enhanced Adaptive Quantization for Channel Sounding Feedback in Wireless Communications," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2024.

[13] B. Zhou, X. Yang, J. Wang, S. Ma, F. Gao, and G. Yang, "A Low-Overhead Incorporation-Extrapolation based Few-Shot CSI Feedback Framework for Massive MIMO Systems," *IEEE Transactions on Wireless Communications*, 2024.

[14] O. Bejarano, E. Magistretti, O. Gurewitz, and E. W. Knightly, "MUTE: Sounding Inhibition for MU-MIMO WLANs," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 135–143, IEEE, 2014.

[15] A. Goldsmith, *Wireless Communications*. Cambridge Univ. Press, 2005.

[16] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[17] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[18] J. Hoydis, S. Cammerer, F. Ait Aoudia, M. Nimier-David, L. Maggi, G. Marcus, A. Vem, and A. Keller, "Sionna," 2022. https://nvlabs.github.io/sionna/.

[19] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz – TR 38.901." https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173, 2026.

[20] K. F. Haque, F. Meneghello, K. Rumman, and F. Restuccia, "m3MIMO: An 8× 8 mmWave Multi-User MIMO Testbed for Wireless Research," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 1922–1929, 2024.

[21] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.

[22] E. Lemaire, L. Cordone, A. Castagnetti, P.-E. Novac, J. Courtois, and B. Miramond, *An Analytical Estimation of Spiking Neural Networks Energy Efficiency*, p. 574–587. Springer International Publishing, 2023.