

Neuromorphic Astronomy: An End-to-End SNN Pipeline for RFI Detection on Hardware

Nicholas J. Pritchard^{1,2}, Andreas Wicenec¹, Richard Dodson¹, Mohammed Bennamoun², and Dylan R. Muir²

¹International Centre for Radio Astronomy Research, University of Western Australia, Australia

²School of Physics, Mathematics and Computing, University of Western Australia, Australia

November 2025

Abstract

Imminent radio telescope observatories provide massive data rates making deep learning based processing appealing while simultaneously demanding real-time performance at low-energy; prohibiting the use of many artificial neural network based approaches. We begin tackling the scientifically existential challenge of Radio Frequency Interference (RFI) detection by deploying deep Spiking Neural Networks (SNNs) on resource-constrained neuromorphic hardware. Our approach partitions large, pre-trained networks onto SynSense Xylo hardware using maximal splitting, a novel greedy algorithm. We validate this pipeline with on-chip power measurements, achieving instrument-scaled inference at 100mW. While our full-scale SNN achieves state-of-the-art accuracy among SNN baselines, our experiments reveal a more important insight that a smaller un-partitioned model significantly outperforms larger, split models. This finding highlights that hardware co-design is paramount for optimal performance. Our work thus provides a practical deployment blueprint, a key insight into the challenges of model scaling, and reinforces radio astronomy as a demanding yet ideal domain for advancing applied neuromorphic computing.

1 Introduction

Increasing demand for neural-network-based Artificial Intelligence (AI) across diverse domains is driving significant growth in data-centre energy usage, underscoring a need for effective, widely deployable, and efficient computing [1, 2]. Current computational approaches meet raw processing requirements, but doing so with lower resource consumption through selective borrowing of concepts from neuroscience remains a promising and active area of research [2]. Radio Frequency Interference (RFI) detection in radio astronomy exemplifies this challenge. RFI consists of anthropogenic radio emission that overwhelms or corrupts the faint cosmic signals telescopes are designed to capture, making accurate and efficient detection critical for the integrity of radio astronomy [3, 4]. RFI is becoming increasingly prevalent, colliding with the increasing sensitivity of future instruments, making RFI detection an existential challenge for the field [5]. Traditional RFI detection algorithms and recent machine-learning-based approaches often treat the problem as an image-segmentation problem suitable for post-observation batch processing [6, 7]. Current operational algorithms require expert calibration, which will not scale to meet the data demands of incoming instruments. Additionally, ANN approaches remain operationally expensive to deploy practically [8], as many model instances are required to run in parallel to service an entire instrument, which produces a spectrogram per pair of antennae (baseline), which for contemporary large interferometric telescopes numbers in the tens of thousands [9].

Spiking Neural Networks (SNNs) offer a compelling alternative to conventional Artificial Neural Networks (ANNs) by drawing their communication mechanism from biology; transmitting discrete, event-based ‘spikes’ rather than continuous weighted sums, with each neuron exhibiting time-varying internal dynamics, lending themselves to temporal information processing [10]. Neuromorphic computers are specialised hardware platforms designed to execute SNN models directly. When the task and input data characteristics align with the time-varying dynamics of SNNs, these systems can yield immense efficiency gains [11] but only through careful consideration of a target task and spiking dynamics [12, 13]. Recent advancements, including reliable backpropagation-like training rules for SNNs [14], and sophisticated open-source tooling [15, 16, 17], have spurred a renaissance in the field. These

advancements have shifted the focus from biological modelling towards addressing practical, high-impact applications such as autonomous driving [18], language modelling [19], audio tagging [20], gesture recognition [21], robotic control [22] and olfactory sensing [23].

Despite these promising developments and the potential of SNNs, the development and deployment of complex, state-of-the-art SNN models onto currently available, often resource-constrained neuromorphic hardware, characterised by finite neuron counts, strict connectivity limitations, and memory constraints, remains a significant challenge [24, 11].

SNNs have previously demonstrated compelling RFI detection performance [25], yet a gap exists in translating high-performing models to real-world neuromorphic hardware. To begin addressing this challenge and exploring the viability of SNNs for this scientific application, we make the following contributions:

1. We develop a Backpropagation-Through-Time (BPTT) trained SNN that achieves high accuracy on a synthetic RFI detection benchmark, surpassing previous algorithmic and SNN baselines and competitive with ANN baselines.
2. We propose ‘maximal splitting’, a novel greedy algorithm designed to shard large pre-trained SNNs onto hardware with specific neuron, synapse, and fan-in constraints.
3. We demonstrate a complete end-to-end pipeline from SNN model training in `snnTorch`, translating to Rockpool using the Neural Intermediate Representation framework (NIR) [17] then deploying split models on the SynSense Xylo 2 neuromorphic processor.

We critically evaluate the maximal splitting method, comparing performance against naive and random splitting baselines, and against small SNN models trained from scratch specifically for direct hardware deployment, and expose a crucial flexibility-versus-efficiency trade-off. Through these efforts, we further validate RFI detection as a demanding yet relevant benchmark for advancing neuromorphic computing research and hardware capabilities, highlighting the need for continued research into hardware-aware training methodologies and deployment pipelines to unlock scalable and high-performing SNN-powered solutions.

2 Related Work

Spiking Neural Networks Spiking Neural Networks (SNNs) represent a class of neural network drawing inspiration from computational principles of biological nervous systems [10, 26]. The core computational unit in SNNs, the spiking neuron, can be modelled in various ways, differing in computational efficiency and biological realism [27]. An increase in complexity makes spiking neurons more expressive [26], but also more difficult to train [14]. The field has spent decades exploring biology-inspired learning rules [28] however, Backpropagation Through Time (BPTT) with surrogate gradients [14] has created somewhat of a watershed moment, as complex SNNs can be trained with increasing reliability and a new generation of neuromorphic hardware promise to unlock previously impossible levels of efficiency in neuron-based processing [2].

Neuromorphic Hardware Aware Training SNNs and neuromorphic computing platforms have developed closely together with their origins rooted in the use of sub-threshold analog circuits to emulate neuron dynamics [29]. Since then, ‘neuromorphic’ hardware has grown to include a wide variety of approaches spanning mimetic biologically-inspired approaches to top-down fully digital architectures [24, 30]. Unlike readily accelerated ANNs, SNNs are often paired with specialised low-precision analog or digital circuits. Contemporary neuromorphic systems impose strict deployment constraints, including weight precision, memory and communication budgets, neuron counts, connectivity limits, and analog hardware mismatches, making practical SNN-based application deployment difficult.

Earlier attempts to deploy SNNs circumvented these issues via ANN-to-SNN conversion strategies [31, 32]. Recent research has moved towards hardware-aware training, where the target system’s constraints are incorporated. These include quantisation-aware training for digital chips like Intel’s Loihi [33], as well as analog-in-the-loop methods running a forward pass in hardware but refining model parameters offline [34]. Finally, emulating larger-than-hardware models sequentially remains a flexible option [35]. Despite progress, most approaches treat SNNs as monolithic models to be mapped entirely to hardware. As neuromorphic hardware and training methodologies mature, new questions emerge around how to co-design scalable, efficient, accurate and deployable models bridging abstract learning paradigms with the realities of hardware-level execution.

RFI Detection RFI detection and mitigation are long-standing critical processes in radio astronomy. Traditional methods rely on fast cumulative-sum-based algorithms such as AOFlagger [6], requiring expert tuning to ‘flag’ contaminated ‘visibilities’ in spectrograms. While operationally effective, manual tuning for specific instruments and observing conditions makes these methods difficult to use for incoming massive radio telescopes such as the Square Kilometre Array (SKA) [9].

The existence of large astronomical datasets and the need for data-driven approaches to RFI detection have encouraged the development of AI-based RFI detection methods. Convolutional Neural Networks (CNNs) and U-Net architectures, in particular, have been extensively explored for this task, treating RFI detection as a two-dimensional semantic segmentation task, just as humans and traditional algorithms do [36, 37, 38, 39, 40]. Employing anomaly detection schemes has also been investigated, utilising auto-encoders [7, 41]. Most recently, authors have explored segment anything networks [42] and vision transformers [43]. These approaches have shown promise, albeit with varying trade-offs in performance, but remain operationally expensive to deploy [8]. The need for real-time RFI detection has spurred the field to explore RFI detection earlier in the signal chain with shorter time-windowed cumulative sum-based approaches [44].

The spectro-temporal nature of radio astronomy visibilities makes it an appealing domain for SNNs [45, 46]. Initial explorations into SNN-based RFI detection adapted an anomaly detection scheme through ANN-to-SNN conversion [47]. Subsequent work focused on directing training SNNs through BPTT by reformulating RFI detection as a time-series segmentation task [25, 48]. Other work explores using liquid state machines for this task with limited success [49]. While still an emerging area, SNN-based methods hold the potential to combine neural networks’ pattern recognition capabilities with low-power, event-driven processing ideal for real-time applications on neuromorphic hardware.

3 Methodology

3.1 RFI Detection with SNNs

We briefly present the preliminaries for RFI detection as a time-series segmentation problem, aligning with existing work in the literature [7, 47, 25]. Radio interferometer array observatories process the raw voltages from antennae into complex-valued ‘visibility’ data $V(v, T, b) \in \mathbb{C}$ varying in frequency, time, and baseline (pair of antennae). RFI detection involves producing a boolean mask of ‘flags’ $G(v, T, b) \in \{0, 1\}$ varying in the same dimensions. The time-series segmentation formulation to RFI detection is given as

$$\mathcal{L}_{\text{sup}} = \min_{\theta_n} (\sum_t^T \mathcal{H}(m_{\theta_n}(E(V(v, t, b), e)), F(G(v, t, b), e))) \quad (1)$$

where θ_n are the parameters of some classifier m , \mathcal{H} is a similarity measure, E is an input encoding function and F is an output encoding function. Both functions introduce a new integer exposure parameter, e , controlling the spike train length for each time step in the original spectrogram. The encoding function results in a time-series of dimensions $(v, T \cdot e, b)$ effectively presenting each time-step in the original spectrogram for an extended period of time e .

We use latency encoding and decoding in this work based on previous works that found this method to be most effective at this task which we present here for completeness [48]. For a given exposure E , input intensities are mapped inversely and linearly from 0 to E . Output decoding interprets spikes before the last exposure step as RFI and all other inputs (including silence) is background. The following equation maps the spike times in the supervised masks:

$$t = \begin{cases} 0 & G(v, t, b) = 1 \\ E & \text{otherwise} \end{cases} \quad (2)$$

The comparison function, \mathcal{H} , is the mean square error of spike timings given by the following function:

$$\mathcal{H}_{\text{latency}} = \sum_e^E \sum_v^V (y_{v,e} - f_{v,e})^2. \quad (3)$$

Fan-in Aware Regularisation We encourage accurate model partitioning by regularising the network to respect the fan-in requirements of the target hardware platform. The penalty term calculates the average excess fan-in, $p_{f_{in}}$, for each neuron in all layers as per,

$$p_{f_{in}} = \frac{1}{f_{in}} \cdot \frac{1}{N} \sum_{i=1}^N \max \left(\sum_{j=1}^M \mathbb{I}(|w_{ij}| > \varepsilon) - f_{in}, 0 \right) \quad (4)$$

where ϵ is a floating-point limit ($1e - 8$), i iterates over each layer in the network, j iterates over each neuron in each layer, \mathbb{I} is the indicator function and f_{in} is the maximum fan-in for the target hardware platform (in this case 63 for the SynSense Xylo 2).

Loss Function Finally, the loss function becomes

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda \cdot p_{f_{in}} \quad (5)$$

where \mathcal{L}_{sup} is the supervised loss function from Equation 1, λ is a hyper-parameter controlling the input of the fan-in penalty and $p_{f_{in}}$ is the fan-in penalty defined in Equation 4. The result is a robust formulation to RFI detection as a time-series segmentation problem with latency encoding and hardware fan-in consideration for SNN training.

3.1.1 Spiking Neuron Model

This work employs a second-order Leaky Integrate and Fire (LiF) neuron model parameterised by τ_{syn} (equivalently $\alpha = e^{(-\Delta t/\tau_{\text{syn}})}$ in `snnTorch`), a decay rate for the synaptic decay, and τ_{mem} (equivalently $\beta = e^{(-\Delta t/\tau_{\text{mem}})}$ in `snnTorch`), a decay rate for the membrane decay. Including both membrane and synaptic decays enables this neuron model to more easily sustain inputs for longer periods, boosting capability in our latency-driven approach [15].

There exists subtle differences in how `snnTorch`, `Rockpool` and `NIR` model the second-order LiF neuron but all are parameterised by the same variables τ_{syn} , and τ_{mem} , in addition to Δt which controls the libraries' simulation granularity. `SnnTorch` models the neuron with the following equation:

$$\begin{aligned} I_{\text{syn}}[t+1] &= e^{\frac{-\Delta t}{\tau_{\text{syn}}}} I_{\text{syn}}[t] + WX[t+1] \\ V_{\text{mem}}[t+1] &= e^{\frac{-\Delta t}{\tau_{\text{mem}}}} V_{\text{mem}}[t] + I_{\text{syn}}[t+1] - R[t] \end{aligned} \quad (6)$$

Where I_{syn} and V_{mem} is the synaptic current and membrane potential respectively, X is the input current, W is an incoming weight and $R[t]$ is a reset mechanism, resetting the neuron's membrane potential by subtraction upon spiking.

`Rockpool` models the neurons with the following equations:

$$\begin{aligned} I_{\text{syn}}^{t+1} &= I_{\text{syn}}^t + S_{\text{in}}(t) + S_{\text{rec}} \cdot W_{\text{rec}} \\ I_{\text{syn}}^{t+1} &= I_{\text{syn}}^t \cdot e^{(-\Delta t/\tau_{\text{syn}})} \\ V_{\text{mem}}^{t+1} &= V_{\text{mem}}^t \cdot e^{(-\Delta t/\tau_{\text{mem}})} \\ V_{\text{mem}}^{t+1} &= V_{\text{mem}}^t + I_{\text{syn}}^t + b + \sigma\zeta(t) \end{aligned} \quad (7)$$

as per `Rockpool`'s implementation [16] where V_{mem} is the membrane voltage potential, I_{syn} is the synaptic current, S_{in} are the incoming spikes, S_{rec} are recurrent spikes, W_{rec} is the current weight, τ_{syn} is the synaptic time-constant, τ_{mem} is the membrane time-constant, b is a bias current, $\zeta(t)$ is a Wigner random noise function, and the neuron membrane is reset by subtraction. While structurally similar to `snnTorch`'s implementation, the change to multiply the updated state variables by a decay rate, and including bias and noise perturbation means there is not a guaranteed direct correspondence between these libraries. This order of operations matches that found in the Xylo hardware. Finally, `NIR` represents a second-order LiF neuron with a Current-based activation LIF model (`CubaLiF`), which is itself composed of a first-order LiF neuron followed by a linear layer and a final leaky integrator discussed most clearly in the original text [17].

Notably, the decay parameters written out by `snnTorch` to `NIR` get read in to `Rockpool` without storing the Δt value which is instead dynamically computed, the result are Δt values which vary within the read network by default, something we manually correct when reading the `NIR` model into `Rockpool`.

3.2 Framework Conversion and Model Architecture

Our methodology, which Figure 1 outlines graphically, bridges the gap between high-level SNN training and deployment on resource-constrained neuromorphic hardware. The core stages involve training a large SNN in `snnTorch` [15], converting it to `NIR` format [17], performing model splitting and adjustments in this format and deploying the models for inference either using `snnTorch` or SynSense Xylo hardware via the `Rockpool` library [16].

The training process involves splitting spectrograms into model-width-sized patches. Each patch is encoded into spikes using latency encoding and a set exposure time per spectrogram-timestep. The resulting spike trains are input into a multi-layer feedforward SNN. We calculate the loss and regularisation, and then backpropagate. After training, the optimised SNN model is exported into NIR format and undergoes splitting using one of three algorithms (detailed later) to shard the model into smaller fixed-width models. Following splitting, we handle these sub-modules in two ways. Firstly, to evaluate accuracy degradation, we load all split sub-modules into snnTorch simultaneously for separate inference on spectrogram patches, allowing for comparison with the originally trained large model. Secondly, for on-chip power evaluation, we load a single sub-model into Rockpool via NIR and deploy it onto real SynSense hardware, running a limited amount of inference to gain indicative power measurements.

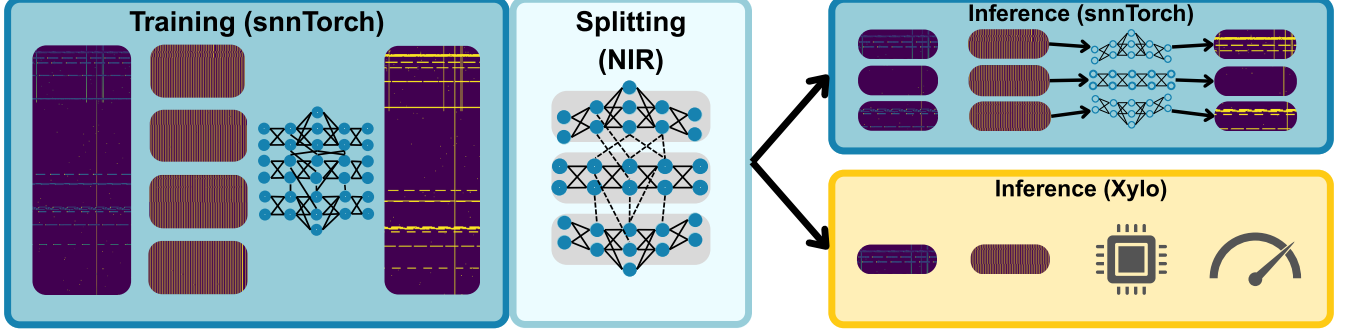


Figure 1: Overall methodology for RFI detection with spiking neural networks trained as a large single model and then split for inference on several neuromorphic chipsets. Spectrograms are split and latency encoded before feeding through the SNN, models are split in NIR format and deployed in snnTorch or to SynSense Xylo hardware for power measurement.

3.2.1 Model Architecture

To simplify the model splitting process and ensure straightforward inference on the target hardware, we exclusively consider fully-connected feedforward SNN architectures in this work. For a given input patch of width W , the network consists of an input layer with W (spiking) neurons. This is followed by an initial hidden layer of $W \cdot 2$ neurons, which then connects to $N - 1$ subsequent hidden layers, each containing H neurons. The final output layer mirrors the input, comprising W neurons, to produce the RFI mask for each time step over e exposure steps. We conducted hyperparameter tuning with Optuna [50] optimising for F1-score including traditional hyper-parameters, snnTorch-specific decay rates (α and β), exposure-time, and fan-in regularisation weighting.

We expect high per-pixel accuracy results owing to the sparsity of RFI in the dataset, and therefore, we opted to optimise for F1-score. Each hyper-parameter trial ran for 25 epochs with an initial learning rate of $1e - 3$ while utilising Lightning’s plateau scheduler, a fixed batch size of 36 for all trials and 10% of the original training set data. Table 1 contains the ranges for each hyper-parameter tested; ‘Num Hidden’ refers to the size of each hidden layer, ‘Num Layers’ refers to the number of hidden layers, including a single scaling layer that is automatically set to be twice the network’s input width and ‘Fan-in-weighting’ refers to the weighting of the fan-in regularisation term. Additionally, α and β are shorthand methods to describe τ_{syn} and τ_{mem} representing $e^{(-\Delta t/\tau_{syn})}$ and $e^{(-\Delta t/\tau_{mem})}$ respectively. All model input widths are tested over the entire parameter range, except the 8-channel input size models, which have a fixed hidden layer size of 64 neurons.

Attribute	Parameter Range
Num Hidden	128, 256, 512, 1024, 2048, 4096
Num Layers	2 - 6
α	0 - 1
β	0 - 1
Exposure	1 - 64
Fan-in-weighting	0 - 0.1

Table 1: Parameter ranges for attributes included in the final hyper-parameter searches.

Table 2 contains the result of the hyper-parameter tuning. We see that the 64-channel input models exhibit the best single-trial performance, followed by the smaller 8-channel and 32-channel models. While performance by the larger model sizes may at first seem disappointing, the data volume used for hyper-parameter training and the lower epoch count likely leaves room for further performance compared to the smaller models since full-channel spectrograms are sliced into model-width-sized sub-spectrograms for training. Therefore, smaller models are trained for more iterations at a fixed number of epochs and data. Furthermore, we see that the smaller models make more use of the fan-in regularisation term, most model widths result in a scale difference between the α and β decay parameters, and the 8-channel model in particular uses an exceptionally large exposure time and similarly scaled α and β parameters for the spiking neurons. Table 3 contains final training parameters.

Model Size	Num Layers	Num Hidden	α	β	Exposure	Fan-in weighting	F1
8	4	64	0.127	0.261	62	0.042	0.945
32	3	2048	0.187	0.073	14	0.063	0.938
64	4	512	0.001	0.198	7	0.064	0.973
128	3	512	0.15	0.06	11	0.074	0.887
256	3	1024	0.166	0.414	21	0.009	0.818
512	4	512	0.413	0.027	2	0.02	0.758

Table 2: Hyper-parameter search using Optuna multi-variate optimisation for the synthetic HERA dataset. Best F1-score and model size is in bold.

Model Size	Num Layers	Num Hidden	α	β	Exp.	Fan-in
8	4	64	0.127	0.261	62	0.042
32	3	2048	0.187	0.073	14	0.063
64	4	512	0.001	0.198	7	0.064
128	3	512	0.15	0.06	11	0.074
256	3	1024	0.166	0.414	21	0.009
512	4	512	0.413	0.027	2	0.02

Table 3: Hyper-parameter search using Optuna multi-variate optimisation for the synthetic HERA dataset. We show the individual trials that performed best in the F1-score.

3.3 Model Splitting Algorithm

Maximal Splitting Maximal splitting takes in as input the original SNN model and a hardware configuration specifying the maximum input and output neurons, neuron fan-in (f_{in}), total number of hidden neurons, number of layers, and maximum total number of connections permitted by a particular hardware platform. The algorithm’s main intuition is to split the output neurons into suitably sized bundles, construct a sub-network that includes the f_{in} most strongly connected inputs to each neuron in each layer, and then cull excess neurons iteratively, selecting the weakest connected neuron across all layers until fan-in, total neuron number and total connection number limits are met. Figure 2.a presents a graphical depiction of this approach, first selecting strongly connected neurons then culling greedily until hardware constraints are met.

Algorithm 1 presents pseudocode for the maximal splitting algorithm. For naive and random splitting strategies, the principal difference is in how **FindLeastImp** proceeds. For a total neuron count N , the algorithm will process each neuron in each iteration and have no more than N iterations providing a worst-case time complexity of $\mathcal{O}(N^2)$ complexity and $\mathcal{O}(N^2)$ memory requirements, to store each possible neuron connection in the worst-case allowing for arbitrary neuron connections between all layers. In practice, however, the total number of iterations will be much lower than N since at most $f_{\text{in}} \cdot N$ neurons will be included in the initial maximal graph and typically $f_{\text{in}} \ll N$.

Naive Splitting Naive splitting works by setting both the input and output neuron bundles simultaneously; naively selecting adjacent m_{width} input and output neurons for each split. The culling process is driven by selecting the geometrically furthest neuron from the input and output neuron, resulting in a tight neuron bundle from input to output depicted in Figure 2.b.

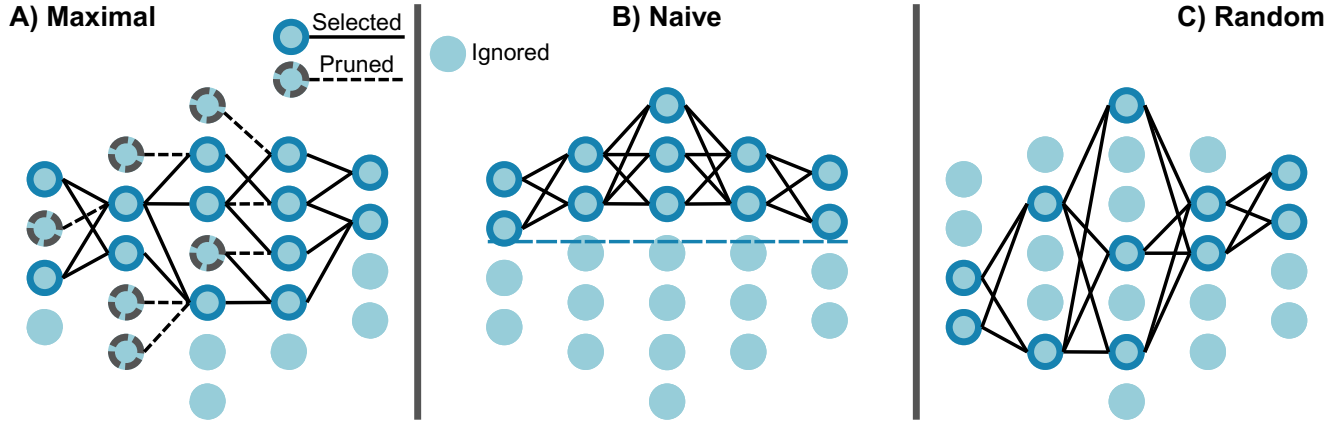


Figure 2: Model splitting approaches. Our model splitting algorithm greedily selects a neuron bundle such that each neuron satisfies the hardware platform restrictions, then iteratively removes neurons until chipset restrictions are met. This is in contrast to a random or naive (geometric) splitting approach also depicted here.

Random Splitting Finally, random splitting selects a random initial sub-network, then randomly culls neurons (including input connections) from each layer in a weighted random choice (weighted by the number of neurons in each layer). The resulting sub-network can exhibit an expected random structure of disparate neurons depicted in Figure 2.c. While straightforward, these model splitting algorithms deliver hardware-compatible sub-graphs suitable for inference on real neuromorphic hardware and represent a first step towards high-level compilation of large SNNs to numerous neuromorphic chipsets.

Algorithm 1 Maximal SNN Splitting

```
1: Procedure MaximalSplit( $m_{\text{orig}}, C_{\text{hw}}$ )
2: output  $\leftarrow []$ 
3: allConns  $\leftarrow$  AllPossConns( $m_{\text{orig}}$ )
4: outBundles  $\leftarrow$  SplitOutputs( $m_{\text{orig}}, C_{\text{hw}}.\text{width}$ )
5: for outB in outBundles do
6:   currCs  $\leftarrow$  allConns.copy
   {Phase 1: Prune Hidden}
7:   while HN(currCs)  $> C_{\text{hw}}.\text{maxN}$  do
8:     nImps  $\leftarrow$  CalcImp(currCs, outB, 'hidden')
9:     leastImpNeuron  $\leftarrow$  FindLeastImp(nImps)
10:    currCs  $\leftarrow$  RemBackwards(leastImpNeuron)
11:   end while
   {Phase 2: Prune Inputs}
12:   while InNeurons(currCs)  $> C_{\text{hw}}.\text{maxInNeurons}$  do
13:     nImps  $\leftarrow$  CalcImp(currCs, outB, 'input')
14:     leastImpIn  $\leftarrow$  FindLeastImp(nImps)
15:     currCs  $\leftarrow$  RemForwards(leastImpIn)
16:   end while
   {Phase 3: Prune Outputs}
17:   while OutConns(currCs)  $> C_{\text{hw}}.\text{maxOutFIn}$  do
18:     nImps  $\leftarrow$  CalcImp(currCs, outB, 'output')
19:     leastImpLast  $\leftarrow$  FindLeastImp(nImps)
20:     currCs  $\leftarrow$  RemBackwards(leastImpLast)
21:   end while
   {Phase 4: Prune Total Connections}
22:   while TConns(currCs)  $> C_{\text{hw}}.\text{maxConns}$  do
23:     nImps  $\leftarrow$  CalcImp(currCs, outB, 'hidden')
24:     leastImpNeuron  $\leftarrow$  FindLeastImp(nImps)
25:     currCs  $\leftarrow$  RemBackwards(leastImpNeuron)
26:   end while
27:   output.append(BuildSubModel(currCs))
28: end for
29: return output
```

4 Experiments

We evaluate our model through several experiments with the Hydrogen Epoch of Reionisation Array (HERA) dataset [7]. First, by training full-sized models of varying input widths. Second, we provide energy and power estimates based on measuring indicative spike-rates of each arbitrary-width model. Third, we split our full-width models into Xylo-compatible sub-modules using our proposed model splitting algorithms and report on detection accuracy. Fourth, we provide power usage measurements for split models on real Xylo hardware. Fifth, we ablate our fan-in regularisation technique, and, finally, we compare model performance against the SoTA in algorithmic, ANN, and SNN-based RFI detection methods, finding our full-width models achieve SoTA detection performance among SNN baselines.

4.1 Training Environment

All experiments were run on single HPE Cray EX nodes featuring an AMD EPYC 7A53 ‘Trento’ 64-core CPU and eight AMD Instinct MI250X GPUs. Original models were trained with snnTorch 0.9.4 [15] using PyTorch 1.13.1+ROCm5.2 and Lightning 2.2.0 [51]. Conversion to Xylo-compatible models was completed with Rockpool 2.9.1 [16] and NIR 1.0.4 [17]. We patched Rokpool’s NIR reading module to fix the dt value to $1E-4$ for CUBA-LiF neurons, as this value is inferred from the τ_{mem} and τ_{syn} values written out to NIR format by default. We trained all models in full precision for 50 epochs using mini-batches of 36 samples, using Adam as the optimiser and a learning rate of $1E-3$. We utilised Lightning’s reduce-lr-on-plateau scheduler (factor 0.5, patience 10) to govern learning-

rate decay. Training-validation splits were deterministic, training data were shuffled each epoch, and testing data order was fixed. Hyper-parameter tuning trials take less than six hours each, full training runs less than 24 each, and all subsequent model splitting, power estimate, and power measurement trials take approximately 15 minutes each. Distributed data-parallel execution spanned all eight GPUs for training, but NIR conversion occurred on a single GPU for final testing. The final model corresponds to the last saved checkpoint. Finally, for power measurement, we utilised a SynSense Xylo Audio 2 development kit for power consumption measurement [52] paired with the Samna 0.45.3 runtime environment.

4.2 Results on HERA Dataset

First, we train models of arbitrary input width as baselines to split into sub-modules. Table 4 contains detection performance results averaged over 10 trials. We observe diminishing returns in accuracy and AUROC beyond 128 input channels, and our 64-channel model achieves the best performance in class-balanced metrics, AUPRC, and F1-score. We also train an 8-channel model suitable for direct deployment on Xylo hardware, providing a baseline for comparison with model splitting techniques. Figure 3 presents an original spectrogram (Figure 3a), and the

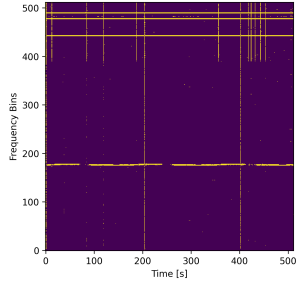
Model Size	Accuracy		AUROC		AUPRC		F1	
8	0.9903	0.001	0.7501	0.019	0.8908	0.024	0.8568	0.032
32	0.9906	0.002	0.9321	0.010	0.8834	0.018	0.8771	0.019
64	0.9986	0.000	0.9880	0.001	0.9833	0.002	0.9827	0.002
128	0.9996	0.000	0.9883	0.003	0.9780	0.006	0.9777	0.006
256	0.9912	0.001	0.9133	0.012	0.8541	0.017	0.8507	0.018
512	0.9897	0.002	0.9311	0.014	0.8746	0.024	0.8718	0.025

Table 4: Detection performance results for full-sized hardware-regularised models of varying sizes on the HERA dataset. Scores are presented as mean and standard deviation over 10 trials, and the best scores are bolded. We see larger, but not the largest models make best use of the available data and perform significantly better than larger or smaller data widths.

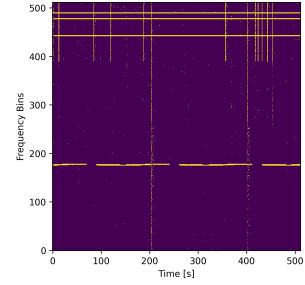
associated ground-truth mask (Figure 3b), and Figure 4 presents example inference for full-width models at varying input widths. We can see that while all models detect the major RFI features, smaller models struggle at the boundaries and produce noisier outputs especially visible in the 32-channel example, Figure 4e.



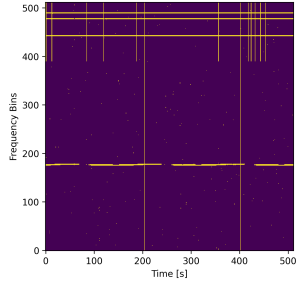
Figure 3: An original full-size spectrogram and its corresponding ground-truth label.



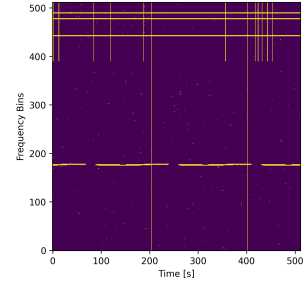
(a) 512 channel inference.



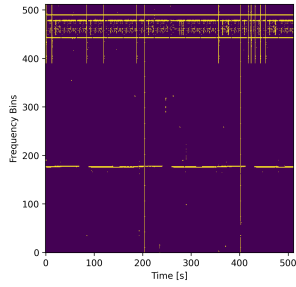
(b) 256 channel inference.



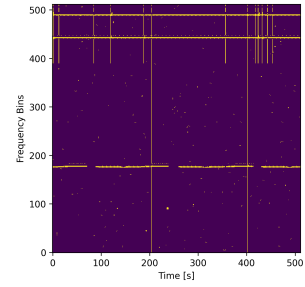
(c) 128 channel inference.



(d) 64 channel inference.



(e) 32 channel inference.



(f) 8 channel inference.

Figure 4: Inference examples across different model sizes. Examples of the original spectrogram in Figure 3 with varying trained model widths and their corresponding hardware reconstructions. We see that in all widths, hardware faithfully reconstructs the output of the software-trained models but that smaller models outperform larger input widths.

4.3 Power Consumption Estimates

Next, we prepare power consumption estimates for our full-width models by running each trained model through the test set, storing spike counts for each layer using a method similar to prior literature [53]. We estimate the synaptic operations for each layer l from the following:

$$SOPs(l) = C_{in} \times C_{out} \times R_s(l) \quad (8)$$

where C_{in} and C_{out} are the input and output sizes of a particular layer and $R_s(l)$ is the spike rate for layer l . We then provide an estimate by considering the spike-based accumulation (AC) operations utilising 40nm CMOS technology at $E_{AC} = 0.9\text{pJ}$ [54]. We use the measured spike-rates of each layer combined with power estimates for accumulate operations to provide a per-spectrogram energy usage estimate, which we can then translate into power measurements at 50MHz (modelling typical Xylo deployment speeds) and also at the minimum clock frequency required for real-time operation in the HERA telescope (an integration time of 3.52 seconds). These values provide a theoretical energy estimate for inference, which we then normalise, factoring in various options for compute time and the exposure time of each model-width’s hyperparameters. We calculate power consumption estimates by first estimating the inference time of a single spectrogram by dividing the number of SOPs by the clockspeed in question. Then, we divide the energy usage estimate for a spectrogram by this value to provide a power estimate given the expected inference time over a single spectrogram.

We multiply the baseline estimate for a model of a particular width to estimate consumption at the full spectrogram channel width (512). Hypothetical real-time frequencies are intentionally set at extremely low levels to provide a conservative lower estimate. Table 5 contains energy and power measurements for each full-size model. Intriguingly, despite the different model scales, the differences in exposure times roughly equalise (except for 32 and 256 channel models) the eventual synaptic operation count, and therefore the overall power consumption. Based on modelling alone, which does not account for device-specific overheads, we expect power usage to range from $< 100\text{mW}$ using the default clock rate down to or below 1 mW at extremely low clock speeds.

Model Size	Exp	Real-Time Hz	SOPs	SE (mJ)	SPE at 50MHz (mW)	SPE at RTF (mW)
8	62	17.61	3.66E+7	0.508	44.4	1.80E-2
32	14	3.98	9.69E+8	11.9	9.79	1.05E-1
64	7	1.99	3.38E+7	0.419	5.08	1.86E-3
128	11	3.13	1.73E+7	0.640	7.39	1.42E-3
256	21	5.97	6.61E+7	60.4	91.5	6.71E-2
512	2	0.568	2.10E+6	0.778	18.5	4.32E-4

Table 5: Energy and power estimates for SNN-based RFI detection. The table details energy per spectrogram (SE) and power estimate (SPE) across various model sizes operating at 50 MHz and a Real-Time Frequency (RTF) dependent on the exposure time used. Results presented as mean over 10 trials and best values are bolded.

4.4 Effect of Model Splitting

We took each trained full-width model and split each into Xylo-compatible sub-modules using one of three splitting algorithms outlined earlier. Table 6 contains results for running all sub-models together to detect RFI over entire spectrograms. We observe that while per-pixel accuracy is not significantly degraded, the performance degradation for AUPRC and F1 Scores is substantial. We also observe that for these regularised models, the best-performing splitting algorithm is maximal splitting up to 128-channel models, indicating that the fan-in limits for each neuron impose an upper limit on the maximum size of the original, unsplit model.

4.5 Power Consumption Measurements

For each full model width, we split 10 regularised models with maximal splitting and measure power consumption over approximately five seconds of inference. We provide measurements at both the default 50MHz clock speed supported by the Xylo 2 board and the minimal supported clock speed of 6.25 MHz. The Xylo HDK provides tooling to measure power consumption, isolating the SNN core itself. Table 7 contains power consumption measurements

Model Size	Splitting Method	Accuracy		AUROC		AUPRC		F1	
32	Maximal	0.9328	0.013	0.5944	0.054	0.2139	0.084	0.1893	0.066
	Naive	0.9437	0.008	0.5195	0.029	0.1044	0.061	0.0974	0.029
	Random	0.9247	0.011	0.5513	0.041	0.1423	0.062	0.1288	0.043
64	Maximal	0.9471	0.009	0.5487	0.042	0.2720	0.111	0.1701	0.084
	Naive	0.9583	0.002	0.5363	0.015	0.3653	0.065	0.1496	0.033
	Random	0.9548	0.005	0.5353	0.035	0.4004	0.072	0.1529	0.085
128	Maximal	0.9583	0.004	0.5655	0.030	0.3167	0.097	0.2081	0.078
	Naive	0.9577	0.002	0.5185	0.008	0.3136	0.042	0.0951	0.016
	Random	0.9552	0.004	0.5252	0.019	0.2455	0.100	0.1126	0.042
256	Maximal	0.5367	0.215	0.3878	0.066	0.1402	0.055	0.0786	0.002
	Naive	0.8684	0.037	0.4953	0.022	0.0803	0.039	0.0795	0.007
	Random	0.8888	0.025	0.4992	0.014	0.0757	0.025	0.0789	0.003
512	Maximal	0.9507	0.010	0.5021	0.006	0.2448	0.211	0.0786	0.002
	Naive	0.9603	0.001	0.5011	0.001	0.4753	0.059	0.0765	0.002
	Random	0.9591	0.001	0.5010	0.002	0.5122	0.024	0.0787	0.002

Table 6: Detection performance results for regularised split models on the HERA dataset. Scores presented as mean and standard deviation over 10 trials and best scores are bolded. Maximal splitting performs best followed by naive and then random approaches for 32, 64 and 128 width models while splitting introduces irrecoverable error for larger model widths.

for varying model sizes. We multiply all measurements and estimates for a single chipset by $\frac{512}{8} = 64$ times to provide values on a per-spectrogram basis. We find that power consumption is extremely low across all splittings, validating our observation of roughly equal Synaptic operations in Table 5.

The spectrogram-scaled power consumption is of particular note, representing the power consumption required to detect RFI in one of several thousand baselines. While a more comprehensive investigation is justified, the potential efficiency gains are immense. Table 8 presents power consumption measurements for all three model

Original Model Size	Exposure	MP at 50MHz (mW)		SP at 50MHz (mw)		MP at 6.25MHz (mW)		SP at 6.25MHz (mW)	
8	62	1.63	0.205	104.5	13.1	0.224	0.001	14.3	0.072
32	14	1.28	0.053	81.7	3.37	0.239	0.007	15.3	0.461
64	7	1.22	0.033	78.1	2.13	0.235	0.007	15.0	0.455
128	11	1.23	0.041	78.4	2.64	0.234	0.004	15.0	0.288
256	21	1.59	0.025	102	1.59	0.237	0.006	15.1	0.359
512	2	1.20	0.019	76.8	1.22	0.239	0.003	15.3	0.184

Table 7: Energy and power measurements for SNN-based RFI detection measured on SynSense Xylo Hardware. Contains model-power (MP) and spectrogram-power (SP) for a single sub-module in varying split sizes.

Results are presented as mean and standard deviation over 10 trials.

splitting algorithms averaged over available trials. We see that power usage is similar regardless of the algorithm used, indicating a consistent and high usage of available resources. We also note the consistency in power usage across all models and splitting algorithms.

4.6 Ablation Study: Hardware-Aware Regularisation

Table 9 compares the best-performing splitting model algorithm for full models with and without hardware-aware regularisation. While improvements are modest, the difference is more significant for the best-performing 64 and 128 models. We also note that the best-performing splitting algorithm is the maximal algorithm up until 128 channel models. This suggests an upper limit imposed by hardware restrictions as the learned dynamics in the larger networks outscale what can be supported within fan-in restrictions.

Model Size	Exposure	Splitting Algorithm	Model Power at 50MHz (mW)		Model Power at 6.25MHz (mW)	
8	62	Maximal	1.63	0.205	0.22	0.001
		Naive	1.63	0.201	0.22	0.002
		Random	1.63	0.202	0.22	0.002
32	14	Maximal	1.28	0.053	0.24	0.007
		Naive	1.28	0.044	0.23	0.006
		Random	1.29	0.038	0.24	0.005
64	7	Maximal	1.22	0.033	0.23	0.007
		Naive	1.23	0.043	0.23	0.007
		Random	1.22	0.026	0.24	0.009
128	11	Maximal	1.23	0.041	0.23	0.004
		Naive	1.22	0.030	0.24	0.010
		Random	1.20	0.027	0.23	0.009
256	21	Maximal	1.59	0.025	0.24	0.006
		Naive	1.59	0.025	0.24	0.007
		Random	1.56	0.019	0.24	0.004
512	2	Maximal	1.20	0.019	0.24	0.003
		Naive	1.22	0.004	0.23	0.009
		Random	1.19	0.055	0.24	0.005

Table 8: Energy and power measurements for SNN-based RFI detection measured on SynSense Xylo Hardware with varying splitting algorithms. The table details power usage for a single sub-module in varying split sizes operating at 50MHz and 6.25 MHz. Measurements are presented as mean and standard deviation over 10 trials.

Size	Reg.	Split	Accuracy		AUROC		AUPRC		F1	
32	Y	Max	0.93	0%	0.59	-2%	0.21	-3%	0.19	-1%
	N	Max	0.93	0%	0.61	-2%	0.24	-3%	0.20	-1%
64	Y	Max	0.95	-1%	0.55	1%	0.27	-5%	0.17	1%
	N	Naive	0.96	-1%	0.54	1%	0.32	-5%	0.16	1%
128	Y	Max	0.96	0%	0.57	2%	0.32	2%	0.21	4%
	N	Max	0.96	0%	0.55	2%	0.30	2%	0.17	4%
256	Y	Rand	0.89	1%	0.50	0%	0.08	-1%	0.08	0%
	N	Rand	0.88	1%	0.50	0%	0.09	-1%	0.08	0%
512	Y	Rand	0.96	0%	0.50	0%	0.51	1%	0.08	0%
	N	Rand	0.96	0%	0.50	0%	0.50	1%	0.08	0%

Table 9: Overall effect of introducing hardware regularisation. Scores are presented as averages and percentage differences.

Tables 10 and 11 present detection performance results for full and split models trained without hardware-aware regularisation, respectively. We see that without regularisation, the best performance comes from the smallest model, where we suspect splitting has the least effect on sub-model dynamics; there are simply fewer changes needed to enforce hardware compatibility. Moreover, the largest model with random splitting provides the best accuracy and AUPRC scores, suggesting that in size comes some element of redundancy. However, performance is still poor in this case.

4.7 Baseline Comparison

Finally, Table 12 compares the performance of our best-performing full-width and split models, our 8-channel Xylo-width model, and prior works. While split models fall short of SoTA performance, the surprising effectiveness of our 8-channel model is promising, and our unsplit 64-channel model achieves SoTA detection performance among SNN baselines and compelling performance against ANN baselines.

Model Size	Accuracy		AUROC		AUPRC		F1		
8	0.9887	0.006	0.7589	0.027	0.8845	0.078	0.8447	0.103	
32	0.9905	0.001	0.9301	0.006	0.8829	0.006	0.8765	0.007	
64	0.9986	0.000	0.9880	0.001	0.9829	0.003	0.9823	0.003	
128	0.9996	0.000	0.9882	0.002	0.9755	0.005	0.9753	0.005	
256	0.9909	0.001	0.9128	0.016	0.8511	0.019	0.8475	0.021	
512	0.9920	0.003	0.9419	0.018	0.9001	0.031	0.8977	0.032	

Table 10: Detection performance results for full-sized unregularised models of varying sizes on the HERA dataset. Scores presented as mean and standard deviation over 10 trials and best scores are bolded.

Model Size	Splitting Method	Accuracy		AUROC		AUPRC		F1	
32	Maximal	0.9272	0.010	0.6109	0.075	0.2444	0.095	0.2036	0.063
	Naive	0.9417	0.012	0.5334	0.060	0.1278	0.089	0.0983	0.034
	Random	0.9213	0.019	0.5326	0.021	0.1174	0.037	0.1090	0.023
64	Maximal	0.9441	0.012	0.5213	0.017	0.1668	0.083	0.1164	0.029
	Naive	0.9584	0.002	0.5413	0.015	0.3213	0.047	0.1551	0.031
	Random	0.9530	0.005	0.5291	0.012	0.2965	0.058	0.1271	0.022
128	Maximal	0.9567	0.003	0.5492	0.019	0.2962	0.083	0.1665	0.050
	Naive	0.9587	0.003	0.5118	0.007	0.3241	0.059	0.0802	0.013
	Random	0.9562	0.006	0.5279	0.027	0.3033	0.093	0.1110	0.048
256	Maximal	0.5972	0.126	0.4093	0.060	0.1301	0.043	0.0782	0.002
	Naive	0.8640	0.043	0.4938	0.031	0.0829	0.035	0.0827	0.017
	Random	0.8772	0.031	0.5018	0.015	0.0866	0.023	0.0798	0.006
512	Maximal	0.9259	0.039	0.4915	0.010	0.2092	0.196	0.0775	0.003
	Naive	0.9591	0.001	0.5006	0.001	0.4692	0.061	0.0786	0.003
	Random	0.9596	0.001	0.5003	0.000	0.5028	0.033	0.0776	0.002

Table 11: Detection performance results for unregularised split models of varying sizes and splitting approaches on the HERA dataset. Scores presented as mean and standard deviation over 10 trials, and best scores are bolded. We see a different trend compared to the regularised model splitting where the best scores either come from a randomly split large model, or maximal split smaller model.

Work	Model	AUROC	AUPRC	F1
<i>Algorithmic Baseline</i>				
Mesarcik et al. [7]	AOFlagger	0.974	0.880	0.873
<i>ANN Baselines</i>				
Vafaei Sadr et al. [39]	R-Net	0.975	0.846	0.846
Yang et al. [40]	RFI-Net	0.973	0.890	0.900
Mesarcik et al. [7]	U-Net	0.975	0.896	0.902
Mesarcik et al. [7]	AutoEncoder	0.981	0.927	0.910
Pritchard et al. [47]	AutoEncoder	0.983	0.940	0.939
van Zyl and Grobler [41]	RFDL	0.994	0.965	0.944
Du Toit, Grobler, and Ludick [8]	ASPP	-	-	0.985
Du Toit, Grobler, and Ludick [8]	RNet-7	-	-	0.989
Du Toit, Grobler, and Ludick [8]	RFI-Net	-	-	0.993
<i>SNN Baselines</i>				
Pritchard et al. [47]	ANN2SNN	0.944	0.920	0.953
Pritchard et al. [48]	BPTT	0.929	0.785	0.761
Pritchard et al. [25]	BPTT	0.996	0.914	0.907
Pritchard et al. [49]	Liquid State Machine	0.842	0.781	0.743
Pritchard et al. [55]	Full Polarisation	0.997	0.960	0.955
<i>This work (SNN)</i>				
This work	Full-8	0.750	0.891	0.857
This work	Split-128	0.566	0.317	0.208
This work	Split-64	0.549	0.272	0.170
This work	Full-64	0.988	0.983	0.983

Table 12: Detection results compared to baselines. Best scores in bold. Our best split model (128-channels, maximally split) lags contemporary models, however our best performing 64-channel offers compelling results.

5 Conclusion and Limitations

This article introduced a pipeline for training and deploying SNNs for the challenging task of RFI detection on resource-constrained neuromorphic hardware. We developed a BPTT-trained SNN achieving high accuracy on a synthetic radio astronomy benchmark dataset, outperforming algorithmic and SNN baselines while remaining competitive with ANN baselines. We then introduced ‘maximal splitting,’ a greedy partitioning algorithm, to deploy our large, pre-trained SNNs onto resource-constrained neuromorphic platforms. This pipeline approach allowed us to train models in `snnTorch`, before performing inference and deployment in SynSense’s Rockpool library and Xylo 2 hardware, making use of the field’s recent push towards interoperability through NIR [17]. Our power usage estimates and measurements demonstrate the immense potential for SNNs and neuromorphic computing to contribute to spectral-temporal data processing in data-intensive fields such as radio astronomy.

While this article demonstrates a promising pipeline for deploying large SNNs for RFI detection onto neuromorphic hardware and achieves SoTA accuracy with the original full-width models, several limitations warrant discussion.

1. *Performance Degradation from Splitting:* The most significant limitation is the performance drop when we split large, high-performing SNNs into hardware-compatible sub-modules. While our ‘maximal splitting’ method provides superior performance to baseline approaches, the split models do not retain SoTA accuracy or match the performance of smaller models trained directly for hardware. This indicates that such an approach may be inherently limited without the building of natural break-points within a network as sub-modules.
2. *Synthetic Benchmark Focus:* The primary SoTA results have been based on the synthetic HERA dataset. While this dataset is valuable and widely used within the literature, real-world astronomy data presents greater variability, more complex noise, and less certainty around ground-truth RFI detection for training. The true generalisability and robustness of our approaches need further validation on observational data.
3. *Hardware Platform Specificity:* We conducted power measurements exclusively on SynSense Xylo hardware. While providing concrete evidence of energy efficiency, this may vary across other neuromorphic architectures with their varying hardware constraints and capabilities.
4. *Feedforward Architecture Constraint:* For simplicity in model splitting and hardware deployment, this work focused on fully-connected feed-forward SNN architectures. This choice may limit learning capacity and the efficacy of model splitting without the ability to pre-define sub-modules.
5. *Temporal Dynamics for Xylo Inference:* While latency encoding was used, achieving sufficiently fine-grained temporal dynamics for optimal inference on the xylo platform after NIR conversion presented challenges. When the SNN is trained around fine-grained timings, any perturbation in the underlying neuron mode could affect overall performance.
6. *Supervised Learning Paradigm:* The core RFI detection method relies on supervised learning, which, while effective, requires labelled datasets. In radio astronomy, unsupervised, self-supervised, or even semi-supervised learning would be more desirable.
7. *Limited Power Consumption Comparisons:* While we present power usage for neuromorphic hardware, a comparison to operational RFI detection pipelines is missing. While comparison is likely to be favourable, without a concrete exploration we cannot make any substantial claims.

Building on these limitations and the successes of our work, several promising avenues for future research emerge:

1. *Advanced Model Splitting and Hardware-Aware Training:* Developing more sophisticated model splitting techniques combined with retraining of sub-models, including pre-defined submodules and additional hardware-aware regularisation offline, would improve performance and efficiency post-splitting.
2. *Validation on Real-World RFI Data:* Extending the evaluation to diverse, real-world radio astronomy data (such as LOFAR, MeerKAT, ASKAP telescopes) is essential to assess the true utility and robustness of proposed methods.
3. *Exploration of Advanced SNN Architectures:* Integrating more complex SNN designs and connection types (e.g. residual, recurrent, and convolutional) into the base models and splitting algorithm approaches could enhance RFI detection and minimise accuracy drops introduced by model splitting.

4. *Optimising Temporal Encoding and Hardware Mapping*: Research into regularisation and compilation techniques to better preserve the fine-grained temporal dynamics during conversion between high-level library platforms and onto hardware could help bridge the performance gap.
5. *Semi, Self, or Un-Supervised Approaches*: Exploring other learning paradigms, such as auto-encoder-based anomaly detection schemes, could produce more robust RFI detection models better suited for real telescope deployment.
6. *Hardware Design Search*: Inverting the model splitting approach to instead search through hardware designs or configurable architectures to find more easily accommodated SNN models.
7. *Alternate Application Domains*: Exploring the applicability of the techniques this work presents to other spectro-temporal data processing domains, such as radar detection or seismic analysis, may extend the reach of SNN-based applications. It is important however to note the potential negative societal impacts of applying SNN-based techniques in dual-use tasks.

Addressing these areas of concern will be key to advancing the practical application of SNNs and neuromorphic computing for RFI detection and more general data-intensive spectro-temporal data processing with neuromorphic computers.

Neuromorphic computing has long been envisioned as a solution for complex in-sensor processing challenges. Radio telescopes, as some of the world’s largest and most complex sensors, present a prime application domain. Casting RFI detection as a time-series problem suitable for real-time SNN processing brings neuromorphic computing closer to the realm of sub-watt supercomputing.

Funding

This work was supported by a Westpac Future Leaders Scholarship, an Australian Government Research Training Program Fees Offset and an Australian Government Research Training Program Stipend.

Roles

N.J.P. was involved with conceptualisation, data curation, funding acquisition, investigation, methodology, software and writing. A.W., R.D. and M.B. provided supervision. R.M. provided resources. All authors reviewed and edited the manuscript.

Data

Dataset is available online: <https://zenodo.org/records/14676274>. Code is also available online <https://zenodo.org/records/17576919>.

References

1. IEA. Energy and AI. en-GB. Tech. rep. International Energy Agency, 2025 Apr. Available from: <https://www.iea.org/reports/energy-and-ai> [Accessed on: 2025 May 14]
2. Muir DR and Sheik S. The road to commercial success for neuromorphic technologies. en. *Nature Communications* 2025 Apr; 16. Publisher: Nature Publishing Group;3586. DOI: 10.1038/s41467-025-57352-1. Available from: <https://www.nature.com/articles/s41467-025-57352-1> [Accessed on: 2025 Apr 22]
3. Thompson AR, Moran JM, and Swenson Jr. GW. *Interferometry and Synthesis in Radio Astronomy*. eng. 3rd ed. 2017. Astronomy and Astrophysics Library. Publication Title: *Interferometry and Synthesis in Radio Astronomy*. Cham: Springer Nature, 2017
4. Nieuwpoort RV van. Towards exascale real-time RFI mitigation. *2016 Radio Frequency Interference (RFI)*. 2016 Oct :69–74. DOI: 10.1109/RFINT.2016.7833534. Available from: <https://ieeexplore.ieee.org/document/7833534> [Accessed on: 2025 Mar 18]
5. Report of the Committee on the Peaceful Uses of Outer Space. EN. Tech. rep. 65. United Nations Office for Outer Space Affairs, 2022 Aug :59. Available from: https://www.unoosa.org/oosa/oosadoc/data/documents/2022/a/a7720_0.html

6. Offringa A. AOFlagger: RFI Software. Astrophysics Source Code Library 2010 :ascl-1010
7. Mesarcik M, Boonstra AJ, Ranguelova E, and van Nieuwpoort RV. Learning to detect radio frequency interference in radio astronomy without seeing it. *Monthly Notices of the Royal Astronomical Society* 2022 Nov; 516:5367–78. DOI: 10.1093/mnras/stac2503. Available from: <https://doi.org/10.1093/mnras/stac2503> [Accessed on: 2023 Mar 13]
8. Du Toit CD, Grobler TL, and Ludick DJ. A comparison framework for deep learning RFI detection algorithms. *Monthly Notices of the Royal Astronomical Society* 2024 May; 530:613–29. DOI: 10.1093/mnras/stae892. Available from: <https://doi.org/10.1093/mnras/stae892> [Accessed on: 2024 Apr 29]
9. Vermij E, Fiorin L, Jongerius R, Hagleitner C, and Bertels K. Challenges in exascale radio astronomy: Can the SKA ride the technology wave? en. *The International Journal of High Performance Computing Applications* 2015 Feb; 29. Publisher: SAGE Publications Ltd STM:37–50. DOI: 10.1177/1094342014549059. Available from: <https://doi.org/10.1177/1094342014549059> [Accessed on: 2024 Oct 28]
10. Trappenberg TP. Fundamentals of computational neuroscience. eng. 2nd ed. Publication Title: Fundamentals of computational neuroscience. Oxford, UK ; Oxford University Press, 2010
11. Schuman CD, Kulkarni SR, Parsa M, Mitchell JP, Date P, and Kay B. Opportunities for neuromorphic computing algorithms and applications. en. *Nature Computational Science* 2022 Jan; 2. Number: 1 Publisher: Nature Publishing Group:10–19. DOI: 10.1038/s43588-021-00184-y. Available from: <https://www.nature.com/articles/s43588-021-00184-y> [Accessed on: 2022 Oct 10]
12. Ottati F, Gao C, Chen Q, Brignone G, Casu MR, Eshraghian JK, and Lavagno L. To Spike or Not To Spike: A Digital Hardware Perspective on Deep Learning Acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2023. Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems:1–1. DOI: 10.1109/JETCAS.2023.3330432. Available from: <https://ieeexplore.ieee.org/document/10309205> [Accessed on: 2023 Nov 07]
13. Dampfhofer M, Mesquida T, Valentian A, and Anghel L. Are SNNs Really More Energy-Efficient Than ANNs? an In-Depth Hardware-Aware Study. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2023 Jun; 7. Conference Name: IEEE Transactions on Emerging Topics in Computational Intelligence:731–41. DOI: 10.1109/TETCI.2022.3214509. Available from: <https://ieeexplore.ieee.org/abstract/document/9927729> [Accessed on: 2025 Mar 23]
14. Neftci EO, Mostafa H, and Zenke F. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine* 2019 Nov; 36. Conference Name: IEEE Signal Processing Magazine:51–63. DOI: 10.1109/MSP.2019.2931595. Available from: <https://ieeexplore.ieee.org/document/8891809> [Accessed on: 2023 Sep 29]
15. Eshraghian JK, Ward M, Neftci E, Wang X, Lenz G, Dwivedi G, Bennamoun M, Jeong DS, and Lu WD. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE* 2023; 111:1016–54
16. Muir DR, Bauer F, and Weidel P. Rockpool Documentaton. 2019 Sep. DOI: 10.5281/zenodo.3773845. Available from: <https://doi.org/10.5281/zenodo.3773845>
17. Pedersen JE, Abreu S, Jobst M, Lenz G, Fra V, Bauer FC, Muir DR, Zhou P, Vogginger B, Heckel K, Urgese G, Shankar S, Stewart TC, Sheik S, and Eshraghian JK. Neuromorphic intermediate representation: A unified instruction set for interoperable brain-inspired computing. en. *Nature Communications* 2024 Sep; 15. Publisher: Nature Publishing Group:8122. DOI: 10.1038/s41467-024-52259-9. Available from: <https://www.nature.com/articles/s41467-024-52259-9> [Accessed on: 2025 May 06]
18. Zhu RJ, Wang Z, Gilpin L, and Eshraghian JK. Autonomous Driving with Spiking Neural Networks. en. *Advances in Neural Information Processing Systems* 2024 Dec; 37:136782–804. Available from: https://proceedings.neurips.cc/paper_files/paper/2024/hash/f7344147dbd1607deac3a7e5f33a23aa-Abstract-Conference.html [Accessed on: 2025 Apr 22]
19. Zhu RJ, Zhao Q, and Eshraghian JK. SpikeGPT: Generative Pre-trained Language Model with Spiking Neural Networks. *arXiv:2302.13939 [cs]*. 2023 Feb. DOI: 10.48550/arXiv.2302.13939. Available from: <http://arxiv.org/abs/2302.13939> [Accessed on: 2023 Mar 31]
20. Bos H and Muir D. Sub-mW Neuromorphic SNN audio processing applications with Rockpool and Xylo. *arXiv:2208.12991*. 2022 Sep. DOI: 10.48550/arXiv.2208.12991. Available from: <http://arxiv.org/abs/2208.12991> [Accessed on: 2024 Oct 22]

21. Massa R, Marchisio A, Martina M, and Shafique M. An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor. *2020 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. 2020 Jul :1–9. DOI: 10.1109/IJCNN48605.2020.9207109. Available from: <https://ieeexplore.ieee.org/document/9207109> [Accessed on: 2025 May 06]
22. Glatz S, Martel J, Kreiser R, Qiao N, and Sandamirskaya Y. Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor. *2019 International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. 2019 May :9631–7. DOI: 10.1109/ICRA.2019.8794145. Available from: <https://ieeexplore.ieee.org/document/8794145> [Accessed on: 2025 Apr 22]
23. Dennler N, Drix D, Rastogi S, Schaik A van, and Schmuker M. Rapid Inference of Geographical Location with an Event-based Electronic Nose. *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*. NICE '22. New York, NY, USA: Association for Computing Machinery, 2022 May :112–4. DOI: 10.1145/3517343.3517381. Available from: <https://dl.acm.org/doi/10.1145/3517343.3517381> [Accessed on: 2025 May 14]
24. Schuman CD, Potok TE, Patton RM, Birdwell JD, Dean ME, Rose GS, and Plank JS. A Survey of Neuro-morphic Computing and Neural Networks in Hardware. arXiv:1705.06963 [cs]. 2017 May. DOI: 10.48550/arXiv.1705.06963. Available from: <http://arxiv.org/abs/1705.06963> [Accessed on: 2022 Dec 09]
25. Pritchard NJ, Wicenc A, Bennamoun M, and Dodson R. Spiking Neural Networks for Radio Frequency Interference Detection in Radio Astronomy. arXiv:2412.06124 [cs]. 2024 Dec. DOI: 10.48550/arXiv.2412.06124. Available from: <http://arxiv.org/abs/2412.06124> [Accessed on: 2025 Jan 07]
26. Maass W. Networks of spiking neurons: The third generation of neural network models. en. *Neural Networks* 1997 Dec; 10:1659–71. DOI: 10.1016/S0893-6080(97)00011-7. Available from: <https://www.sciencedirect.com/science/article/pii/S0893608097000117> [Accessed on: 2023 Mar 21]
27. Gerstner W and Kistler WM. Spiking neuron models: Single neurons, populations, plasticity. Cambridge university press, 2002
28. Taherkhani A, Belatreche A, Li Y, Cosma G, Maguire LP, and McGinnity TM. A review of learning in biologically plausible spiking neural networks. *Neural Networks* 2020 Feb; 122:253–72. DOI: 10.1016/j.neunet.2019.09.036. Available from: <https://www.sciencedirect.com/science/article/pii/S0893608019303181> [Accessed on: 2025 May 14]
29. Mead C. Neuromorphic electronic systems. *Proceedings of the IEEE* 1990 Oct; 78. Conference Name: Proceedings of the IEEE:1629–36. DOI: 10.1109/5.58356
30. Frenkel C, Bol D, and Indiveri G. Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence. *Proceedings of the IEEE* 2023 Jun; 111. Conference Name: Proceedings of the IEEE:623–52. DOI: 10.1109/JPROC.2023.3273520. Available from: <https://ieeexplore.ieee.org/document/10144567> [Accessed on: 2024 Mar 13]
31. Esser SK, Appuswamy R, Merolla P, Arthur JV, and Modha DS. Backpropagation for Energy-Efficient Neuromorphic Computing. *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. Available from: <https://proceedings.neurips.cc/paper/2015/hash/10a5ab2db37feedfdeaab192ead4ac0e-Abstract.html> [Accessed on: 2022 May 05]
32. Rueckauer B, Lungu IA, Hu Y, Pfeiffer M, and Liu SC. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience* 2017; 11. Publisher: Frontiers Media SA:682
33. Davies M, Srinivasa N, Lin TH, Chinya G, Cao Y, Choday SH, Dimou G, Joshi P, Imam N, Jain S, Liao Y, Lin CK, Lines A, Liu R, Mathaikutty D, McCoy S, Paul A, Tse J, Venkataramanan G, Weng YH, Wild A, Yang Y, and Wang H. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 2018 Jan; 38. Conference Name: IEEE Micro:82–99. DOI: 10.1109/MM.2018.112130359. Available from: <https://ieeexplore.ieee.org/document/8259423/>
34. Cramer B, Billaudelle S, Kanya S, Leibfried A, Grübl A, Karasenko V, Pehle C, Schreiber K, Stradmann Y, Weis J, Schemmel J, and Zenke F. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences* 2022 Jan; 119. Publisher: Proceedings of the National Academy of Sciences:e2109194119. DOI: 10.1073/pnas.2109194119. Available from: <https://www.pnas.org/doi/abs/10.1073/pnas.2109194119> [Accessed on: 2024 May 24]

35. Arnold E, Spilger P, Straub JV, Müller E, Dold D, Meoni G, and Schemmel J. Scalable network emulation on analog neuromorphic hardware. English. *Frontiers in Neuroscience* 2025 Feb; 18. Publisher: Frontiers. DOI: 10.3389/fnins.2024.1523331. Available from: <https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2024.1523331/full> [Accessed on: 2025 Jul 29]
36. Akeret J, Chang C, Lucchi A, and Refregier A. Radio frequency interference mitigation using deep convolutional neural networks. en. *Astronomy and Computing* 2017 Jan; 18:35–9. DOI: 10.1016/j.ascom.2017.01.002. Available from: <https://www.sciencedirect.com/science/article/pii/S2213133716301056> [Accessed on: 2022 Sep 20]
37. Mosiane O, Oozeer N, Aniyani A, and Bassett BA. Radio Frequency Interference Detection using Machine Learning. en. *IOP Conference Series: Materials Science and Engineering* 2017 May; 198. Publisher: IOP Publishing:012012. DOI: 10.1088/1757-899X/198/1/012012. Available from: <https://doi.org/10.1088/1757-899X/198/1/012012>
38. Vos EE, Luus FPS, Finlay CJ, and Bassett BA. A Generative Machine Learning Approach to RFI Mitigation for Radio Astronomy. English. *2019 IEEE 29TH INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING (MLSP)*. IEEE International Workshop on Machine Learning for Signal Processing. Backup Publisher: IEEE ISSN: 2161-0363 Type: Proceedings Paper. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE, 2019
39. Vafaei Sadr A, Bassett BA, Oozeer N, Fantaye Y, and Finlay C. Deep learning improves identification of Radio Frequency Interference. *Monthly Notices of the Royal Astronomical Society* 2020 Oct; 499:379–90. DOI: 10.1093/mnras/staa2724. Available from: <https://doi.org/10.1093/mnras/staa2724> [Accessed on: 2023 Aug 16]
40. Yang Z, Yu C, Xiao J, and Zhang B. Deep residual detection of radio frequency interference for FAST. *Monthly Notices of the Royal Astronomical Society* 2020 Feb; 492:1421–31. DOI: 10.1093/mnras/stz3521. Available from: <https://doi.org/10.1093/mnras/stz3521> [Accessed on: 2023 Aug 16]
41. van Zyl DJ and Grobler TL. Remove First Detect Later: a counter-intuitive approach for detecting radio frequency interference in radio sky imagery. *Monthly Notices of the Royal Astronomical Society* 2024 May; 530:1907–20. DOI: 10.1093/mnras/stae979. Available from: <https://doi.org/10.1093/mnras/stae979> [Accessed on: 2024 Oct 23]
42. Deal D and Jagannathan P. Segmenting RFI Using Meta’s Segment Anything Model. en. *Research Notes of the AAS* 2024 Oct; 8. Publisher: The American Astronomical Society:257. DOI: 10.3847/2515-5172/ad84fa. Available from: <https://dx.doi.org/10.3847/2515-5172/ad84fa> [Accessed on: 2024 Oct 31]
43. Ouyang X, Dreuning H, Mesarcik M, and Nieuwpoort RV van. Hierarchical vision transformers for RFI mitigation in radio astronomy. en. *Bariloche, Argentina*, 2024 Oct
44. Mehrabi A, Dennler N, Bethi Y, Schaik Av, and Afshar S. Real-Time Anomaly Detection Using Hardware-based Unsupervised Spiking Neural Network (TinySNN). *2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE)*. ISSN: 2163-5145. 2024 Jun :1–8. DOI: 10.1109/ISIE54533.2024.10595773. Available from: <https://ieeexplore.ieee.org/abstract/document/10595773> [Accessed on: 2025 Jan 21]
45. Scott NM. Evolving Spiking Neural Networks for Spatio- and Spectro- Temporal Data Analysis: Models, Implementations, Applications. en. PhD thesis. Auckland University of Technology, 2015. Available from: <https://hdl.handle.net/10292/10601> [Accessed on: 2024 Apr 05]
46. Kasabov N, Scott NM, Tu E, Marks S, Sengupta N, Capecci E, Othman M, Doborjeh MG, Murli N, Hartono R, Espinosa-Ramos JI, Zhou L, Alvi FB, Wang G, Taylor D, Feigin V, Gulyaev S, Mahmoud M, Hou ZG, and Yang J. Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications. *Neural Networks. Special Issue on "Neural Network Learning in Big Data"* 2016 Jun; 78:1–14. DOI: 10.1016/j.neunet.2015.09.011. Available from: <https://www.sciencedirect.com/science/article/pii/S08933608015001860> [Accessed on: 2024 Mar 15]
47. Pritchard NJ, Wicenc A, Bennamoun M, and Dodson R. RFI Detection with Spiking Neural Networks. en. *Publications of the Astronomical Society of Australia* 2024 Apr :1–11. DOI: 10.1017/pasa.2024.27. Available from: <https://www.cambridge.org/core/journals/publications-of-the-astronomical-society-of-australia/article/rfi-detection-with-spiking-neural-networks/994EA6FDFA18B7799D7F0552264111E6> [Accessed on: 2024 Apr 05]

48. Pritchard NJ, Wicenec A, Bennamoun M, and Dodson R. Supervised Radio Frequency Interference Detection with SNNs. *2024 International Conference on Neuromorphic Systems (ICONS)*. 2024 Jul :102–9. DOI: 10.1109/ICONS62911.2024.00023. Available from: <https://ieeexplore.ieee.org/document/10766534> [Accessed on: 2024 Dec 06]
49. Pritchard NJ, Wicenec A, Bennamoun M, and Dodson R. Advancing RFI-Detection in Radio Astronomy with Liquid State Machines. *2025 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. 2025 Jun :1–7. DOI: 10.1109/IJCNN64981.2025.11229299. Available from: <https://ieeexplore.ieee.org/document/11229299> [Accessed on: 2025 Nov 17]
50. Akiba T, Sano S, Yanase T, Ohta T, and Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019
51. Falcon W and The PyTorch Lightning team. PyTorch Lightning. 2019 Mar. DOI: 10.5281/zenodo.3828935. Available from: <https://github.com/Lightning-AI/lightning> [Accessed on: 2024 Oct 18]
52. AG S. Xylo™: Ultra-low power neuromorphic chip | SynSense. en-US. 2022 Mar. Available from: <https://www.synsense.ai/products/xylo/> [Accessed on: 2024 Oct 22]
53. Barchid S, Mennesson J, Eshraghian J, Djéraba C, and Bennamoun M. Spiking neural networks for frame-based and event-based single object localization. *Neurocomputing* 2023 Sep :126805. DOI: 10.1016/j.neucom.2023.126805. Available from: <https://www.sciencedirect.com/science/article/pii/S0925231223009281> [Accessed on: 2023 Sep 18]
54. Horowitz M. 1.1 Computing’s energy problem (and what we can do about it). *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. ISSN: 2376-8606. 2014 Feb :10–14. DOI: 10.1109/ISSCC.2014.6757323. Available from: <https://ieeexplore.ieee.org/document/6757323> [Accessed on: 2025 Feb 03]
55. Pritchard NJ, Wicenec A, Dodson R, and Bennamoun M. Polarization-Inclusive Spiking Neural Networks for Real-Time RFI Detection in Modern Radio Telescopes. en. *URSI Radio Science Letters* 2025; 7. DOI: 10.46620/25-0006. Available from: <https://doi.org/10.46620/25-0006> [Accessed on: 2025 Sep 22]