# Streaming, End-to-End, Multi-Rate, Automatic Modulation Classification with SNNs

Peter Brass
peter.brass@strath.ac.uk
University of Strathclyde
Glasgow, United Kingdom

Carmine Clemente
carmine.clemente@strath.ac.uk
University of Strathclyde
Glasgow, United Kingdom

Gaetano Di Caterina
gaetano.di-caterina@strath.ac.uk
University of Strathclyde
Glasgow, United Kingdom

## Abstract

Neuromorphic radio-frequency (RF) sensing is attractive for always-on edge devices, but RF data-rate interfaces and conventional preprocessing can offset activity-dependent hardware advantages. To mitigate this constraint on deployment hardware and the related resource demands during training, we present a simulated multi-rate spiking neural network architecture that progressively compresses streaming in-phase/quadrature (I/Q) time series data. Relative to single-rate operation, this yields 50- and 40-fold reductions in rate-normalised neuron updates and GPU memory use, respectively, while achieving high-SNR validation F1 above 0.9 on a synthetic ten-family narrowband automatic modulation classification task. We additionally present supporting training techniques, including a persistent-state protocol that jointly controls state handling and data progression. Under synthetic receiver/front-end impairments absent from the training waveforms, configurations using this protocol show mean validation F1 degradation of about 0.03, compared with 0.14 for corresponding reset-state configurations. A supervised class-similarity hidden-layer loss provides additional gains primarily at high SNR or when combined with the persistent-state protocol.

## CCS Concepts

• **Computing methodologies → Bio-inspired approaches**.

## Keywords

neuromorphic RF sensing, spiking neural networks, automatic modulation classification, multi-rate temporal compression, persistent-state training, supervised class-similarity loss, rate-normalised update density

## 1 Introduction

The activity-dependent power consumption of neuromorphic devices is attractive for always-on radio-frequency (RF) edge sensors [16]. Real-time RF sensing can require acquisition at rates from several to hundreds of megasamples per second, creating compute, memory, and data-transfer pressure in downstream processing [13]. Digital neuromorphic radar studies show that timestep count can materially affect accuracy and computational cost, and that non-neuromorphic preprocessing and interface overhead can undermine the neuromorphic advantage [18]. Reported neuromorphic RF preprocessing studies also show that implementing conventional stages neuromorphically is not automatically efficient: a neuromorphic Fast Fourier Transform (FFT) and cell-averaging constant-false-alarm-rate (CA-CFAR) variants show significant hardware-dependent latency and energy trade-offs [11, 19].

RF tasks can therefore span a wide data–feature–decision rate hierarchy: in-phase/quadrature (I/Q) samples arrive at the acquisition rate, local features emerge over short groups of samples, and class decisions aggregate those features over much longer intervals. Learning relationships across these scales is difficult for recurrent spiking neural networks (SNNs), which inherit limitations of recurrent neural networks (RNNs) in long-range sequence modelling, while adding non-differentiable binary spiking activations [17]. Similarly, the combination of high acquisition rates and long decision intervals constrains backpropagation through time (BPTT) training, because GPU memory must retain intermediate neural states and activations at every unrolled timestep [3, 12, 20].

To bridge this hierarchy, we propose a multi-rate SNN in which successive layers operate at progressively reduced local update rates, so each deeper-layer timestep covers an increasingly long interval of the original I/Q sequence. At each rate-reducing layer boundary, a neuron population applies learned temporal-convolution weights to accumulated ordered outputs from the preceding faster layer, combines them with earlier recurrent spikes in its local rate domain, and performs one Leaky Integrate-and-Fire (LIF) update. Regular temporal sparsity is enforced in the wider, deeper layers of the network, allowing them to integrate features over long signals without updating once per input sample and reducing the state histories retained in GPU memory during BPTT.

The feasibility of this approach is evaluated on non-cooperative Automatic Modulation Classification (AMC), an RF edge-sensing task in which a receiver identifies modulation structure from observed signals without transmitter cooperation [14, 21]. The controlled benchmark uses synthetic I/Q signals and signal-to-noise ratio (SNR) curricula to test whether temporal compression can retain discriminative amplitude, phase, spectral, and timing structure relevant to cognitive radio and electronic surveillance [14].

Two additional training mechanisms were intended to make long-sequence training more efficient and stable. The persistent-state protocol carries network state across boundaries between training steps, using it as informed initialisation, and uses the relationship between subsequent stimuli as a performance-based feedback variable to progressively increase task difficulty. Separately, supervised class-similarity hidden-layer losses use training metadata to shape hidden-layer behaviour and encourage class-specific temporal representations.

## 2 Background

Deep-learning AMC has an established non-spiking radio-frequency machine-learning (RFML) literature using convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers [14]. Neuromorphic RF (NMRF) processing often applies

SNNs after spectrogram or range–Doppler pipelines, and related work has addressed spectrum surveillance and electronic-support tasks such as low-probability-of-intercept (LPI) waveform detection [6, 8, 15]. These examples establish the feasibility of NMRF systems, but many still rely on image-like RF representations, narrow detection objectives, or fixed observation windows rather than continuous minimally processed I/Q streams.

At the opposite end of the task scale, an ultra-low-power 1.1 nW on–off keying (OOK) receiver performs a simpler RF detection task with only two or three Leaky Integrate-and-Fire (LIF) neurons [9]. Recent direct-I/Q SNN AMC work sits between image-like NMRF pipelines and this simple detector, pushing natively neuromorphic processing toward more complex RF tasks.

Guo et al.'s direct-I/Q approach uses the similarity between Sigma-Delta analog-to-digital converters (ADCs) and LIF neurons to treat oversampled ADC output as spike-encoded time-domain AMC data [4]. Instead of reconstructing or decimating the spike stream into a slower, higher-resolution conventional sample representation for digital signal processing (DSP), fixed windows of oversampled spike-like binary ADC output are fed to an SNN implemented on a field-programmable gate array (FPGA). This aligns the shorter-scale oversampled ADC dimension with SNN simulation time while representing the longer radio frame spatially across input-layer neurons.

Happek et al.'s direct-I/Q study applies sample-synchronous SNN processing to 500-timestep AMC windows [5]. Its analysis focuses on digital and analog neuromorphic implementations, while spike encoding, initialisation, timing, and state propagation across windows remain less optimised. Together, these examples reduce conversion overhead or preserve sample-synchronous processing within fixed windows, but still leave open how to reduce real-time neuron-update burden while preserving state over an always-on stream.

## 3 Methods

The method targets five linked constraints:

- non-differentiable spike events;
- long I/Q sequences;
- mismatch between RF sample rate and neuromorphic update rate;
- state discontinuities introduced by interval training; and
- weak supervision of deep hidden spike trains.

Training is performed in discrete time, offline, using GPU simulation with snnTorch for neuron models and surrogate-gradient descent for optimization [3, 12]. LIF neurons provide simple stateful temporal dynamics with low model complexity [3]. A single neuron is modeled by its membrane potential $u \in \mathbb{R}$ and its spiking output $s_t \in \{0, 1\}$. The membrane potential is updated every timestep $t$ by a weighted input $\mathbf{s}_t$ and decay according to a constant $\beta \in [0, 1]$ according to:

$$u_t = \beta u_{t-1} + \mathbf{w} \cdot \mathbf{s}_t - s_{t-1}\theta. \tag{1}$$

The threshold $\theta$ is subtracted after the previous spike $s_{t-1} \in \{0, 1\}$, and binary spike emission is determined by

$$s_t = \begin{cases} 0, & \text{if } u_t < \theta, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

**Table 1: Multi-rate architecture values shared by all configurations.**

| Setting | Value |
|---|---|
| Hidden widths | 8, 16, 32, 64, $128 \times 3$, $256 \times 3$ |
| Output layer | 10 family neurons |
| Compression | $\rho = 2$ in layers 0–8; $\rho = 1$ in layers 9–10 |
| Temporal buffers | $k_f^{(l)} = k_r^{(l)} = 2$ |

Equations (1) and (2) are depicted in the right-hand "Neurons" box of Figure 1.

For layers after the input, $\mathbf{s}_t$ contains spikes emitted by the previous layer at the current layer update and buffered same-layer recurrent spikes from earlier updates. The first layer is an exception: two I/Q pseudo-neurons provide real-valued I and Q samples rather than binary spikes. All inputs are weighted by $\mathbf{w}$ as described in Figure 1, and synaptic weights plus LIF parameters $\beta$ and $\theta$ are optimized during training.

Surrogate-gradient descent addresses the non-differentiable threshold in (2) by replacing it with a differentiable backward-pass approximation [12]. This makes gradient-based learning possible for binary-spike forward dynamics, but approximation error can accumulate with depth and sequence length [3]. BPTT follows causal state transfer from one timestep to the next, but requires intermediate membrane, spike, and buffer histories to remain in GPU memory [3, 12, 20]. Truncated BPTT (TBPTT) addresses this memory pressure by splitting simulation into intervals, applying backpropagation within each interval, and carrying only forward state to the next interval [3, 20]. This truncation is a compromise for AMC because discriminative signal structure may be brief and separated over longer timescales than a single training interval [17].

### 3.1 Multi-Rate SNN Architecture

The multi-rate architecture addresses the mismatch between fast RF samples and slower neuromorphic update rates. On current and future hardware platforms, more neurons that update less frequently may be easier to map than a smaller population forced to follow the input rate [7]. Delta-sigma front ends provide one example of mixed-rate RF-to-spike processing, motivating architectures that bridge an AMC signal's data rate and feature size without first returning to a conventional uniformly sampled representation [4].

Layers of the proposed architecture progressively convert high-rate I/Q samples into lower-rate spike trains, as shown in Figure 1, by accumulating multiple successive outputs of previous layers before updating. The compression factor $\rho$, forward buffer length $k_f$, and recurrent buffer length $k_r$ describe how each layer trades update rate, temporal context, and recurrent state. The forward compression and buffer parameters correspond to stride and kernel size in a Temporal Convolutional Neural Network (TCNN), while also representing potential interfaces between hardware blocks or clock-rate domains.

Using the layer widths $N^{(l)}$ and compression factors in Table 1, with cumulative compression $C^{(l)} = \prod_{m=0}^{l} \rho^{(m)}$, the rate-normalised LIF update density is $U = \sum_l N^{(l)}/C^{(l)} = 25.0195$ per

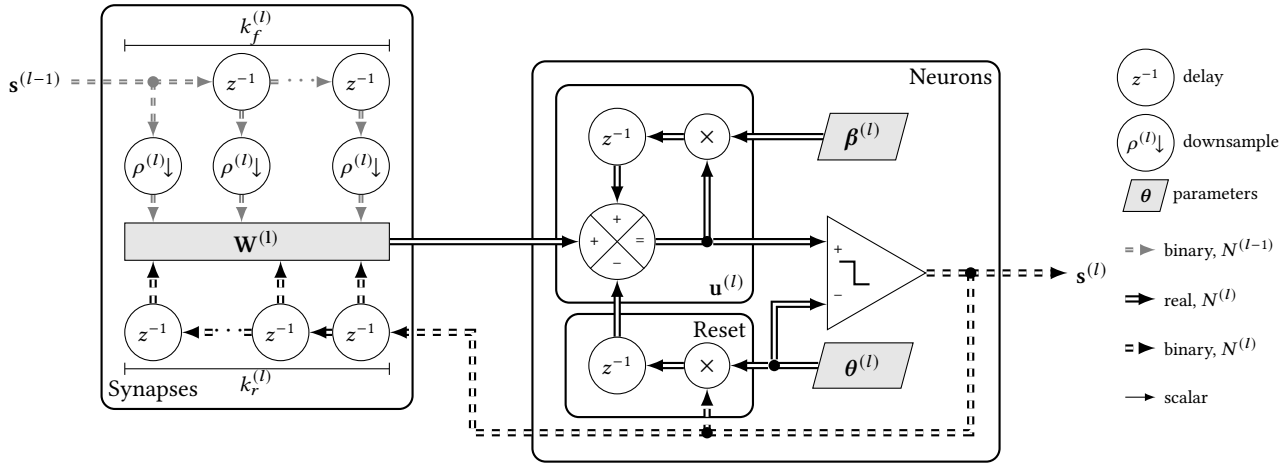


**Figure 1: One multi-rate SNN layer buffers previous-layer spikes ($\mathbf{s}^{(l-1)}$), downsamples by its compression factor $\rho^{(l)}$, combines $k_f^{(l)}$ buffered spiking outputs from the previous layer with $k_r^{(l)}$ recurrent historical emissions of its own spikes ($\mathbf{s}^{(l)}$). Previous layer values are shown in grey and have a width corresponding to the number of neurons in the previous layer ($N^{(l-1)}$). Current layer values are shown in black, with dashed lines representing binary spikes and solid lines representing real valued currents or potentials. Current layer connections carry one value for each current layer neuron and therefore have width ($N^{(l)}$).**

incoming I/Q sample pair. The corresponding single-rate architecture requires $U_{\text{single}} = \sum_l N^{(l)} = 1{,}282$ updates per pair, so temporal compression reduces update density by 51.2×.

This likely represents significantly more processing than the demonstrated task requires; however, a sufficiently large neuron population may still be needed to perform it. For comparison, Happek et al.'s shallow direct-I/Q SNN has 1024 + 512 + 4 LIF neurons. Taking a classification after the first 100 of 500 timesteps, where reported confidence curves reach their maxima, gives $U = (1024 + 512 + 4)100/500 = 308$, about 12.3× higher [5].

Having neurons running at multiple rates throughout the network allows a deliberate distribution of neuron-update resources across timescale and processing stage. In this case only eight actual LIF neurons run at the fastest internal rate, $f_{\text{IQ}}/2$, while the wider 128–256-neuron layers operate after 32×–512× temporal compression.

Reducing the neuron-update density also reduces the GPU resource usage required for TBPTT. At batch 64, a matched synthetic-input training-step profile reduced peak allocated GPU memory from 6,469 MiB without temporal compression to 159 MiB with 512× compression (40.6×). This permits larger batches or longer training intervals, both of which improve training stability.

## 3.2 Persistent State Training

Persistent state training addresses the discontinuity created when long signals are split into short training intervals. Stateful Long Short-Term Memory (LSTM) training has shown that recurrent state can persist across subsequence boundaries while TBPTT still trains on short slices [10]. For the multi-rate SNN, this is especially relevant because deep layers receive very few updates per interval after temporal compression, making their initial membrane and buffer state disproportionately important. Persistent state can therefore act as informed initialisation for later intervals or subsequent signals, reducing the need for deep layers to reconstruct context from only a small number of new updates.

Conventionally, when a batch of samples used for training is complete it is replaced with a new batch, and in the case of stateful networks that state is reinitialised. For a recurrent SNN this entails clamping the membrane potentials to 0 and discarding buffered spikes. If this is not done, then the state persists, each initialised network within the batch gains meaningful identity, and the data presented to it on subsequent optimiser steps becomes a choice in designing the training protocol.

For the Persist and Both configurations (see Table 2), each GPU-resident batch is processed over successive TBPTT intervals and replayed after a resident waveform ends. Presentation then restarts at sample zero while optimiser updates continue and each row retains its membrane potential and forward/recurrent spike buffers. Membrane potentials are clipped to $[-1, 1]$ because repeated signal–state trajectories can otherwise grow faster than parameter updates can recover.

Resident inputs change at two levels. Batches are replaced when the proportion of rows that achieve a multilabel exact match classification, $p$, exceeds a threshold. Before the whole batch advances, $\lfloor p^2 B \rfloor$ signals that were exactly matched by network predictions are randomly permuted among existing rows without moving the network state. The squared rule increases turnover smoothly as $p$ rises while ensuring that the number selected cannot exceed the number of exact-matched rows.

Base and Sim instead fully zero membrane and spike-buffer state at each signal boundary, advance to a new batch after the previous batch is presented once, and do not permute that batch while in use.

**Table 2: Controls used by the four evaluated training configurations.**

| Control | Base | Persist | Sim | Both |
|---|---|---|---|---|
| Buffered Spikes | reset | retain | reset | retain |
| Membrane clip | $[0, 0]$ | $[-1, 1]$ | $[0, 0]$ | $[-1, 1]$ |
| Batch advance | 1 signal | $p > 0.618$ | 1 signal | $p > 0.618$ |
| Swap fraction | 0 | $p^2$ | 0 | $p^2$ |
| Class-sim. weight | 0 | 0 | $10^{-4}$ | $10^{-4}$ |

## 3.3 Training Objective

The training objective combines output classification, hidden spike-rate, and optional class-similarity components as $\mathcal{L} = \sum_j \lambda_j \mathcal{L}_j^{q_j}$. For sample $b$ and class $c$, let $z_{bc}$ be the one-hot label and let $S_{bct}^{(10)}$ denote output spikes across $T$ timesteps. Classification is the mean squared error between the output spike rate and a class-dependent target rate:

$$\begin{aligned} \hat{y}_{bc} &= \frac{1}{T} \sum_t S_{bct}^{(10)}, \\ r_{bc}^* &= r_{\text{off}} + z_{bc} \left( r_{\text{on}} - r_{\text{off}} \right), \\ \mathcal{L}_{\text{cls}} &= \frac{1}{BC} \sum_{b,c} \left( \hat{y}_{bc} - r_{bc}^* \right)^2 . \end{aligned} \tag{3}$$

The labelled and off-target rates are $r_{\text{on}} = 1.0$ and $r_{\text{off}} = 0.2$, respectively. The non-zero off-target rate keeps all output neurons active while the higher on-target rate encourages the labelled class to spike.

Hidden spike-rate losses address silent or saturated hidden populations that can arise from surrogate-gradient error accumulation or exploding and vanishing gradients. For hidden spikes $s_{bnt}$, with $N$ hidden neurons and $T$ timesteps per batch, the bulk sparsity term controls total average activity:

$$\mathcal{L}_{\text{sparsity}} = \left[ \frac{1}{BNT} \sum_{b,n,t} s_{bnt} - s_{\text{sparsity}}^* \right]^2 . \tag{4}$$

A low target $s_{\text{sparsity}}^*$ penalises dense population-level spiking. The per-neuron activity term prevents individual hidden neurons from becoming silent by penalising deviation from a higher target rate:

$$\mathcal{L}_{\text{activity}} = \frac{1}{N} \sum_n \left[ \frac{1}{BT} \sum_{b,t} s_{bnt} - s_{\text{activity}}^* \right]^2 . \tag{5}$$

Together, the spike-rate losses encourage sparse but active hidden representations. Both are applied to layers 0–9, with $s_{\text{sparsity}}^* = 0.0$ and $s_{\text{activity}}^* = 0.8$; each uses $\lambda = 10^{-4}$.

Contrastive hidden-layer losses address weak supervision of deep temporal spike patterns when learning depends only on the output classification loss. They use class labels to shape internal network behaviour after the cost of offline backpropagation has already been paid. For hidden-layer spikes $S_{bnt}^{(l)}$, class fingerprint $A_{cnt}^{(l)}$ is formed by centring spikes at 0.5 and averaging over batch samples of each class:

$$A_{cnt}^{(l)} = \frac{1}{|\{b : c \in \mathbf{y}_b\}|} \sum_{b:c \in \mathbf{y}_b} \left( S_{bnt}^{(l)} - 0.5 \right). \tag{6}$$

Cosine similarity between class fingerprints is then penalised for off-diagonal class pairs:

$$\mathcal{L}_{\text{class-sim}}^{(l)} = \frac{1}{N^{(l)} C^2} \sum_{n,c \neq d} \cos \left( \mathbf{A}_{cn}^{(l)}, \mathbf{A}_{dn}^{(l)} \right). \tag{7}$$

Centering at 0.5 makes the similarity term invariant to spike/non-spike encoding inversion and helps decouple it from spike-rate control, while layer-specific weights balance class separation against sparsity. The simple time-aligned fingerprint has low computational cost, but assumes class-specific patterns are aligned and is therefore less suitable for offset-sensitive early layers. This term is disabled in the "Base" and "Persist" configurations and enabled in the "Sim" and "Both" configurations; Sim is therefore the class-similarity-only configuration. For Sim and Both, it is applied to layers 4–8 with relative layer weights 0.125, 0.25, 0.5, 1, 0.25. Its aggregate coefficient is given in Table 2.

## 3.4 Dataset and Task

A modulation-family classification task using isolated narrowband TorchSig signals was selected as a controlled first test of streamed-I/Q AMC. On related TorchSig data, such reduced-complexity family classification is near-saturated by conventional deep learning, whereas specific-modulation classification remains substantially more difficult [1]. The selected task therefore provides a development step towards individual-modulation classification and the location and classification of multiple signals in wideband recordings. TorchSig narrowband synthetic I/Q signals assume one isolated signal with bandwidth 12.5–25% of sample rate and centre frequency near DC [1]. The present setting uses 4,096 I/Q samples and ten family labels, deliberately removing within-family discrimination. Each example occupies the full sample range; training uses 655,360 signals and validation uses 2,560 signals with equal family representation. Training uses a coarse easy-to-hard clean-SNR curriculum (54, 27, 14, 7, 0 dB) with additive white Gaussian noise (AWGN) at the active SNR.

Training and validation use the same generation procedure, save for the presence of simulated front-end and receiver impairments in validation data. Validation every 1,024 steps uses clean and impaired data; after 2,048 steps without improvement on the matching SNR validation loss, the current stage ends and training advances to the next SNR. Validation sets network state to zero at each signal boundary and uses top-output multiclass decisions over all output timesteps. The receiver/front-end-impaired condition applies TorchSig level-2 impairments at load time [2]. These include IQ imbalance/DC offset, carrier-phase rotation, spectral inversion, LO phase noise, LO frequency drift, and quantisation. Before AWGN, each clean waveform is unit-power normalised to remove SNR distributions as learnable discriminative features.

# 4 Results

Figure 2 reports F1 across clean and synthetically impaired validation data after each SNR curriculum stage. All configurations reach high F1 at 7 dB and above on recently trained clean data, with several clean current-SNR cases exceeding 0.9. At the 7 dB current-training stage, the clean versus receiver/front-end-impaired F1 pairs are 0.904/0.827 for Base, 0.917/0.889 for Persist, 0.851/0.756

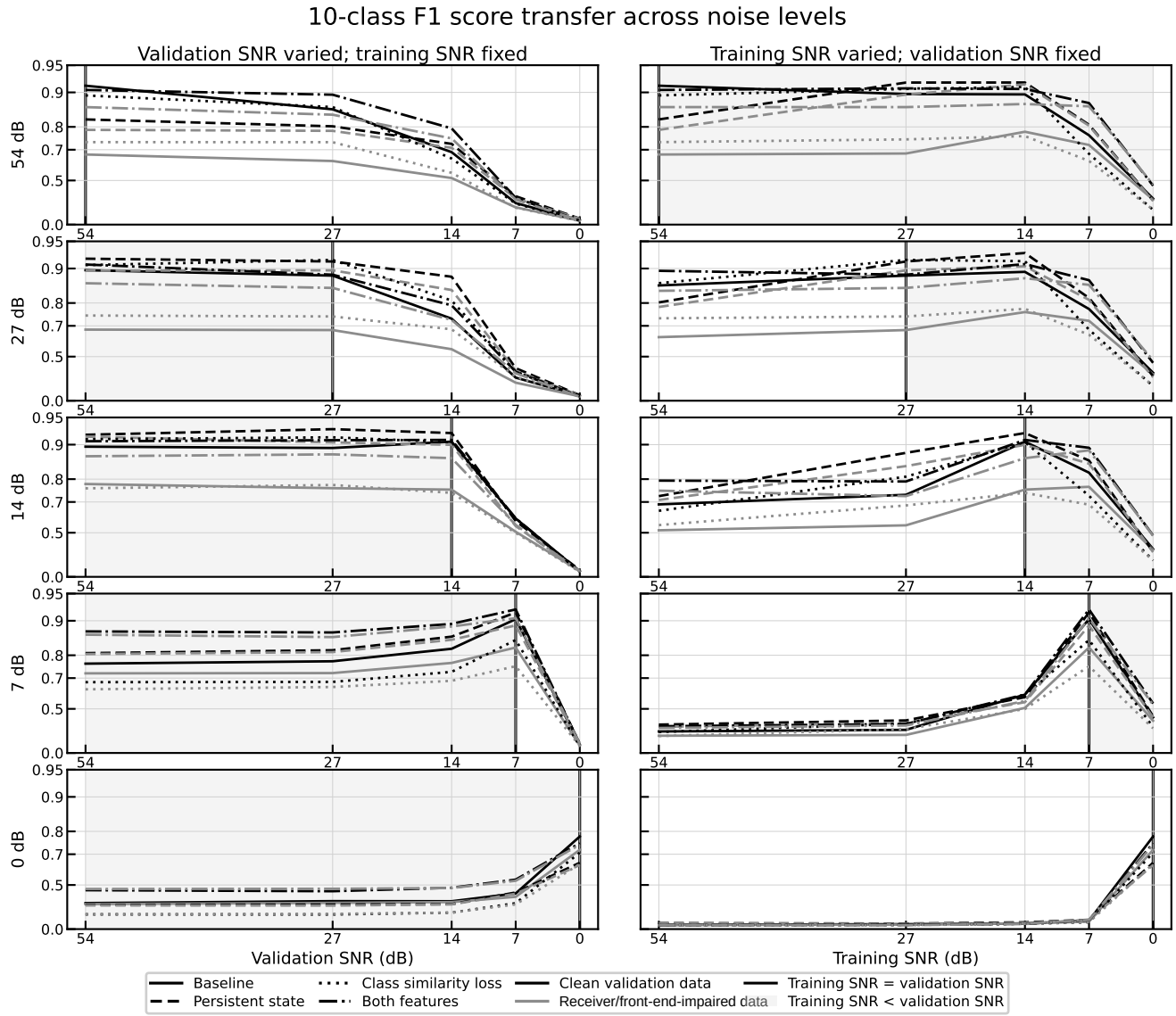


**Figure 2: F1 scores across validation conditions and completed clean-SNR curriculum stages for the four training configurations. The two columns provide complementary views of the same training–validation SNR matrix. Each left-hand panel fixes the training SNR of the completed curriculum stage and varies validation SNR; each right-hand panel fixes validation SNR and varies the completed training stage. The $54 \rightarrow 27 \rightarrow 14 \rightarrow 7 \rightarrow 0$ dB curriculum therefore progresses down the left column and from left to right within each right-hand panel. Vertical markers indicate equal training and validation SNR, while shading indicates conditions in which training SNR is lower than validation SNR.**

for Sim, and 0.923/0.906 for Both. At the 0 dB current-training stage, clean F1 reaches approximately 0.653–0.780, so useful low-SNR performance requires explicit low-SNR training.

Across the two matched configuration pairs, mean clean-to-receiver/front-end-impaired F1 degradation over the five diagonal current-training-SNR stages is 0.146 for Base versus 0.024 for Persist, and 0.130 for Sim versus 0.029 for Both. Persist and Both therefore show approximately fivefold smaller degradation than their reset-state counterparts. Both is strongest in several impaired, unseen-SNR, and earlier-stage retention cases, while Persist remains competitive in current-SNR cells. Class-similarity loss can help at high SNR and when combined with persistence, but used alone it underperforms the baseline on previously seen data at the 7 and 0 dB stages.

## 5 Discussion and Conclusion

The high-SNR F1 results show that surrogate-gradient backpropagation can train a temporally compressed multi-rate SNN on long-sequence time-domain I/Q data. The evidence remains bounded to synthetic ten-family TorchSig AMC with zero-initialised validation boundaries. Each configuration used one training seed, with the five clean curriculum stages requiring approximately 42–77 hours per configuration; confidence intervals across independently trained models are therefore unavailable. Validation loss determined when each SNR curriculum stage ended, and the same validation data supplied the reported metrics, so no independent held-out test result is reported. Evaluation under broader controlled impairments and on over-the-air recordings would test generalisation beyond the synthetic validation distribution.

Within this controlled setting, high-SNR F1 exceeds 0.9, explicit 0 dB training gives useful low-SNR performance, and Persist and Both show approximately fivefold smaller mean clean-to-receiver/front-end-impaired F1 degradation than their corresponding reset-state configurations. This establishes that state handling and data scheduling are part of the learning problem rather than an implementation detail, and suggests that assumptions inherited from independent batched ANN training can be mismatched to temporal SNN dynamics. The class-similarity loss remains conditional: it helps high-SNR performance and retention with persistence, but underperforms alone at low SNR. Further work should prioritise multi-seed replication, component-wise ablation of the persistent-state protocol, matched uncompressed and external baselines, and state-preserved validation. In particular, the hardware implications of reduced neuron-update density must be validated and quantified on a specific plausible NMRF processing deployment. Subject to these evaluations, multi-rate temporal compression, persistent-state training, and conditional hidden-layer supervision suggest that adapting training techniques and architectures to better fit the context of SNNs and RF processing can assist in training and inference of neuromorphic systems on minimally processed RF streams.

## Acknowledgments

This work was supported by Leonardo UK Ltd.

## References

[1] Luke Boegner, Manbir Gulati, Garrett Vanhoy, Phillip Vallance, Bradley Comar, Silvija Kokalj-Filipovic, Craig Lennon, and Robert D. Miller. 2022. Large Scale Radio Frequency Signal Classification. https://doi.org/10.48550/ARXIV.2207.09918

[2] Luke Boegner, Garrett Vanhoy, Phillip Vallance, Manbir Gulati, Dresden Feitzinger, Bradley Comar, and Robert D. Miller. 2022. Large Scale Radio Frequency Wideband Signal Detection & Recognition. https://doi.org/10.31219/osf.io/cvus7

[3] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. 2023. Training Spiking Neural Networks Using Lessons From Deep Learning. *Proc. IEEE* 111, 9 (Sept. 2023), 1016–1054. https://doi.org/10.1109/JPROC.2023.3308088

[4] Wenzhe Guo, Kuilian Yang, Haralampos-G. Stratigopoulos, Hassan Aboushady, and Khaled Nabil Salama. 2023. An End-To-End Neuromorphic Radio Classification System with an Efficient Sigma-Delta-Based Spike Encoding Scheme. *IEEE Transactions on Artificial Intelligence* (2023), 1–14. https://doi.org/10.1109/TAI.2023.3306334

[5] Leon Happek, Jen-Tse Huang, Alejandro Garcia Gener, Rainer Leupers, and Melvin Galicia. 2024. Spiking Neural Networks for Signal Classification with Digital and Analog Neuromorphic Systems. In *2024 International Joint Conference on Neural Networks (IJCNN)*. 1–7. https://doi.org/10.1109/IJCNN60899.2024.10650824

[6] Alex Henderson, Steven Harbour, Chris Yakopcic, Tarek Taha, David Brown, Justin Tieman, and Garrett Hall. 2023. Spiking Neural Networks for LPI Radar Waveform Recognition with Neuromorphic Computing. In *2023 IEEE Radar Conference (RadarConf23)*. 1–6. https://doi.org/10.1109/RadarConf2351548.2023.10149797

[7] Sebastian Höppner, Bernhard Vogginger, Yexin Yan, Andreas Dixius, Stefan Scholze, Johannes Partzsch, Felix Neumärker, Stephan Hartmann, Stefan Schiefer, Georg Ellguth, Love Cederstroem, Luis A. Plana, Jim Garside, Steve Furber, and Christian Mayr. 2019. Dynamic Power Management for Neuromorphic Many-Core Systems. *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, 8 (Aug. 2019), 2973–2986. https://doi.org/10.1109/TCSI.2019.2911898

[8] Xinqiao Jiang, Hongtu Xie, Zheng Lu, and Jun Hu. 2023. Energy-Efficient and High-Performance Ship Classification Strategy Based on Siamese Spiking Neural Network in Dual-Polarized SAR Images. *Remote Sensing* 15, 20 (Jan. 2023), 4966. https://doi.org/10.3390/rs15204966

[9] Zalfa Jouni, Thomas Soupizet, Siqi Wang, Aziz Benlarbi-Delai, and Pietro M. Ferreira. 2023. RF Neuromorphic Spiking Sensor for Smart IoT Devices. *Analog Integrated Circuits and Signal Processing* (July 2023). https://doi.org/10.1007/s10470-023-02164-w

[10] Alexander Katrompas and Vangelis Metsis. 2022. Enhancing LSTM Models with Self-attention and Stateful Training. In *Intelligent Systems and Applications*, Kohei Arai (Ed.). Springer International Publishing, Cham, 217–235. https://doi.org/10.1007/978-3-030-82193-7_14

[11] Javier López-Randulfe, Nico Reeb, Negin Karimi, Chen Liu, Hector A. Gonzalez, Robin Dietrich, Bernhard Vogginger, Christian Mayr, and Alois Knoll. 2022. Time-Coded Spiking Fourier Transform in Neuromorphic Hardware. *IEEE Trans. Comput.* 71, 11 (Nov. 2022), 2792–2802. https://doi.org/10.1109/TC.2022.3162708

[12] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. 2019. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine* 36, 6 (Nov. 2019), 51–63. https://doi.org/10.1109/MSP.2019.2931595

[13] Seckin Oncu, Mehmet Karakaya, Yaser Dalveren, Ali Kara, and Mohammad Derawi. 2024. Real-Time Radar Classification Based on Software-Defined Radio Platforms: Enhancing Processing Speed and Accuracy with Graphics Processing Unit Acceleration. *Sensors* 24, 23 (Jan. 2024), 7776. https://doi.org/10.3390/s24237776

[14] Shengliang Peng, Shujun Sun, and Yu-Dong Yao. 2022. A Survey of Modulation Classification Using Deep Learning: Signal Representation and Data Preprocessing. *IEEE Transactions on Neural Networks and Learning Systems* 33, 12 (Dec. 2022), 7020–7038. https://doi.org/10.1109/TNNLS.2021.3085433

[15] Ali Safa, André Bourdoux, Ilja Ocket, Francky Catthoor, and Georges G. E. Gielen. 2022. On the Use of Spiking Neural Networks for Ultralow-Power Radar Gesture Recognition. *IEEE Microwave and Wireless Components Letters* 32, 3 (March 2022), 222–225. https://doi.org/10.1109/LMWC.2021.3125959

[16] Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, Prasanna Date, and Bill Kay. 2022. Opportunities for Neuromorphic Computing Algorithms and Applications. *Nature Computational Science* 2, 1 (Jan. 2022), 10–19. https://doi.org/10.1038/s43588-021-00184-y

[17] Matei-Ioan Stan and Oliver Rhodes. 2024. Learning Long Sequences in Spiking Neural Networks. *Scientific Reports* 14, 1 (Sept. 2024), 21957. https://doi.org/10.1038/s41598-024-71678-8

[18] Bernhard Vogginger, Felix Kreutz, Javier López-Randulfe, Chen Liu, Robin Dietrich, Hector A. Gonzalez, Daniel Scholz, Nico Reeb, Daniel Auge, Julian Hille, Muhammad Arsalan, Florian Mirus, Cyprian Grassmann, Alois Knoll, and Christian Mayr. 2022. Automotive Radar Processing With Spiking Neural Networks: Concepts and Challenges. *Frontiers in Neuroscience* 16 (2022).

[19] Bernhard Vogginger, Chen Liu, Felix Kreutz, Florian Kelber, Seifeddine Saadani, Klaus Knobloch, Alfred Höß, Cyprian Grassmann, and Christian Mayr. 2025. Automotive Radar Processing with Neuromorphic Hardware: A Case Study from the KI-ASIC Project. In *2025 16th German Microwave Conference (GeMiC)*. 630–633. https://doi.org/10.23919/GeMiC64734.2025.10979154

[20] P.J. Werbos. 1990. Backpropagation through Time: What It Does and How to Do It. *Proc. IEEE* 78, 10 (Oct. 1990), 1550–1560. https://doi.org/10.1109/5.58337

[21] Fuxin Zhang, Chunbo Luo, Jialang Xu, Yang Luo, and Fu-Chun Zheng. 2022. Deep Learning Based Automatic Modulation Recognition: Models, Datasets, and Challenges. *Digital Signal Processing* 129 (Sept. 2022), 103650. https://doi.org/10.1016/j.dsp.2022.103650