ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Classification of radio signals on a neuromorphic processor in space

Master Thesis

Massimo Bortone

April 8th, 2019

Advisors: Prof. Giacomo Indiveri, Prof. Tobi Delbruck, Raphaela Kreiser

Institute of Neuroinformatics, ETH Zürich

Abstract

An increased demand for more efficient and autonomous communication systems in space applications is driving the adoption of artificial intelligence and machine learning methods to solve various tasks, such as adaptive coding and resource-allocation on software-defined radios. At the same time, recent advances in neuromorphic engineering allow for low-power computation with real-time on-line learning capabilities. This work considers the feasibility of an automatic modulation recognition system implemented on a neuromorphic processor using reservoir computing.

Contents

1 Introduction

1.1 Next-generation space communication systems

Future human and robotic space missions, as well as by the various commercial applications of CubeSats[17], demand more efficient and higher-throughput space communication systems. In both domains, large amounts of data needs to be transmitted between communication links in a reliable manner. Further, the increasing computational performance of general purpose processors (GPPs) and field programmable gate arrays (FPGAs) has allowed the adoption of software-defined radios (SDRs) in space applications. NASA is therefore studying the use of artificial intelligence and machine learning algorithms to increase efficiency and reduce complexity of next-generation space communication systems. The aim is to enable higher orders of system automation, while also guaranteeing the high level of reliability that space exploration requires.

The efficient allocation of resources on SDRs is a problem that can greatly benefit from artificial intelligence and machine learning algorithms. In this context, Hackett et al.[7] have developed a cognitive engine that combines a multi-objective reinforcement learning (RL) algorithm with a deep neural-network (NN) to be used as a radio-resource-allocation controller. The reinforcement learning neural network (RLNN) replaces standard lookup-tables, which contain information about various parameters that control the performance of the communication system. Experts carefully compile them to maintain quasi-error-free transmission.

The RLNN implements a multi-objective optimization scheme, which allows the SDR to select the optimal action tuplet for a given channel state. The action tuplet is a set of parameters that define the performance of the communication channel. This includes the modulation and coding scheme, the filter roll-off factor, the symbol rate and transmitter power output. The mapping of environmental states to action tuplets which gives the best performance (highest reward) is the goal of the reinforcement learning algorithm.

In order to improve the online performance of the reinforcement learning algorithm, Ferreira et al. proposed equipping it with a neural network to allows exploration of action tuplets in a virtual environment[5]. The NN thus provides an approximation for the communication environment in terms of the RL agent experience, the previously chosen actions and the respective rewards achieved so far. The virtual exploration reduces the time the RL agent would spend on testing action tuplets that yield to unstable or bad communication channels, by predicting the performance of action tuplets all at once and rejecting those that are below a user defined threshold.

The approximation of the communication environment provided by the NN can be improved when information regarding the modulation and coding scheme in use is available. In this case the RLNN learns to map the most reliable modulation scheme to different environmental conditions. Space communications systems aboard the ISS, for example, have to deal with effects such as multipath-fading and shadowing due to the structures surrounding the antennas. To increase throughput and efficiency, Downey et al. have studied the use of variable[4] and adaptive[3] modulation and coding methods and tested them on the SCaN experimental testbed aboard the ISS[18]. These methods perform forward modelling and inference of the communication link dynamics and have been shown to improve the autonomy of the communication systems.

1.2 Neuromorphic modulation recognition

The aim of this project is to study the feasibility and to assess the performance of a real-time autonomous modulation classification (AMC) system implemented on neuromorphic hardware.

Augmenting the controller developed in [7] with such a system would allow the SDR to adapt the modulation scheme used to encode and decode information according to changes in the environment and the policy learned by the RLNN. This should increase the transmission efficiency and therefore the reliability of the communication channel.

The currently best-performing AMC system is based on a convolutional neural network (CNN) developed by O'Shea[15]. This system outperforms previous systems implemented using traditional machine learning methods achieving a classification accuracy of 87.4% across a signal to noise ratio (SNR) ranging from -20dB to +20dB. The CNN however requires a runtime of $\approx 1\,\mathrm{s}$ for training on the RadioML dataset[14] and has a classification runtime of $\approx 0.5\,\mathrm{s}$.

Although the computational power of CNNs has been widely proven across multiple application domains, these networks still require large amounts of data and power resources when being trained. In space applications, power resources can be a limiting factor.

Secondly, CNNs are more exposed to critical failure due to damage caused to the computational hardware by cosmic rays, thermal excursion, micrometeorites and other phenomena. Such events, can for example damage memory blocks on a microcontroller that store the weights of the CNN layers, leading to dropout of multiple units in the network. In this case, retraining on the whole dataset is necessary to accommodate the loss of computational units. However, the limited resources of satellites make this option rather difficult in most cases.

Brain-inspired neuromorphic hardware lends itself as an appropriate computational substrate for the realization of edge computing applications. In these scenarios the energy consumption is critical and a high level of system robustness and autonomy are required. Neuromorphic devices such as the DYNAPs [13] and the Reconfigurable On-Line Learning System (ROLLS) [16] implement spiking neural networks using mixed signal analog electronics and are thus considered to be ultra-low power inference engines. ROLLS also supports bio-plausible synaptic plasticity rules for online learning capabilities, which would allow the AMC system to continuously improve its performance using the feedback from the RLNN controller. Neural plasticity also offers a mechanism for the system to recover from damage to units in the computational substrate, which does not involve retraining it on the whole dataset.

In this work, I consider the task of modulation recognition as a multi-class classification problem of sampled radio signals, where the label is the modulation scheme used to encode information on a carrier radio signal. The data is obtained from the 2016 version of the RadioML dataset. Details about it are presented in the following chapter.

1.3 Encoding information in radio signals

In this section I briefly describe how information is encoded and decoded in a radio signal using modulation processes. I also present the differences between the most commonly used analog and digital modulation schemes.

In order to communicate via radio signals, a protocol that specifies rules on how to encode information onto a carrier signal needs to be specified. This protocol is a modulation process and the carrier signal is a periodic electromagnetic waveform with a given frequency and amplitude.

The carrier signal is emitted by an encoder and is received by a decoder after propagating through the environment. It can be represented as a sinusoidal function

$$c(t) = A(t)\sin(2\pi f_c t + \varphi(t)) \tag{1.1}$$

where the carrier frequency f_c is usually much higher than the frequency of the modulating input signal $m(t)$.

In analog modulation, $m(t)$ is a continuous signal and is modulated onto the carrier signal by varying the amplitude (A), frequency (f_c) or phase (φ) of $c(t)$.

Digital modulation is typically referred to as *keying* and in this case $m(t)$ is a stream of bits. This digital information is mapped to a finite set of M symbols (alphabet) at a symbol rate f_s. Each symbol in the alphabet thus represents a message on N bits and defines the state of the carrier signal. Depending on the property of the carrier used to encode information the different digital modulation schemes are categorized as:

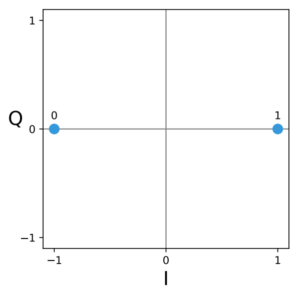
- **phase-shift keying (PSK)**: a set of M phases is used
- **frequency-shift keying (FSK)**: a set of M frequencies is used
- **amplitude-shift keying (ASK)**: a set of M amplitudes is used
- **quadrature-amplitude modulation (QAM)**: a set of M phases and amplitudes is used.

Thermal noise and differences in the electronic components between the transmitter and receiver introduce distortions in the transmitted radio signal such as temporal shifting, scaling, mixing and spinning. Additionally, reflections from buildings and other structures in the environment cause multi-path fading due to destructive interference and phase shifting. All of these effects lead to a general loss of quality of communication, both in digital as well as in analog modulation.

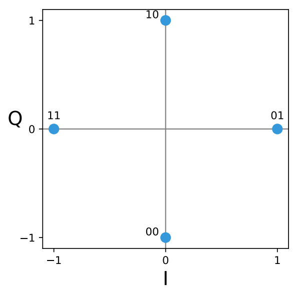
At any given time instant, a modulated radio signal $s(t)$ can be represented as a point in the complex plane

$$s(t) = A(t)e^{i\theta(t)} \tag{1.2}$$

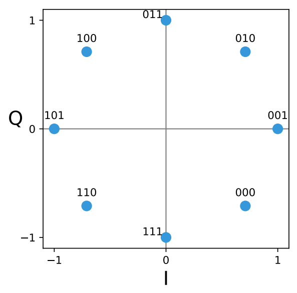
where $\theta(t)$ is the phase angle of the modulated signal. By taking the real and imaginary part of $s(t)$, the signal can be projected onto a basis of orthogonal sinusoidal functions. In this representation the radio signal is decomposed as



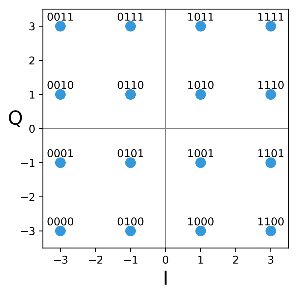
(a) BPSK



(b) QPSK



(c) 8PSK



(d) QAM16

Figure 1.1: Constellation diagrams for 4 different digital modulation process. The blue dots represent the baseband symbols used to encode digital information (string of 0s and 1s above the dots) on to a carrier radio signal. Diagrams (a)-(c) belong to phase-shift-keying schemes, which encode information by altering the phase of the carrier signal. The quadrature-amplitude-modulation in (d) also changes the amplitude.

$$s(t) = I(t)\cos(2\pi f_c t) - Q(t)\sin(2\pi f_c t) \tag{1.3}$$

where $I(t) = \Re\left[A(t)e^{i\theta(t)}\right]$ is called the in-phase component and $Q(t) = \Im\left[A(t)e^{i\theta(t)}\right]$ is the quadrature component.

Using this representation, the alphabets of PSK and QAM digital modulations can be visualized in a constellation diagram. Figure 1.1 shows such diagrams for 4 different modulations.

In a digital modem device a stream of bits is converted into a radio signal in 6 steps, starting with the mapping of words of N bits to the according baseband symbols $d[k]$. The conversion of digital information into an analog signal is then achieved by weighting a Dirac comb function with the baseband symbols

$$x(t) = \sum_{k=-\infty}^{+\infty} d[k]\delta(t - kT_s) \tag{1.4}$$

where k is the symbol index and T_s is the symbol spacing, meaning the time duration of each symbol in the analog signal. After that, the Dirac comb function is convolved with a pulse shaping filter $g(t)$

$$u(t) = g(t) \star x(t) = \sum_{k=-\infty}^{+\infty} d[k]g(t - kT_s). \tag{1.5}$$

The pulse shaping filter used in the RadioML dataset is a root-raised-cosine (RCC) filter, which spans a set of r baseband symbols left and right of the current position. A delay $t_0 = rT_s$ is therefore introduced by the filter.

The final step in the modulation of digital information onto the carrier signal consists in taking the real and imaginary parts of $u(t)$ to obtain the I and Q components

$$I(t) = \Re\left[u(t)\right] \tag{1.6}$$
$$Q(t) = \Im\left[u(t)\right]. \tag{1.7}$$

The transmitted signal $s(t)$ is then obtained by combining the components as defined by Eq. 1.3. An example of IQ-data using digital QPSK modulation is shown in Figure 1.2.

At the receiving end of the communication channel the incoming signal $s(t)$ is projected onto the basis functions to recover the I and Q components. These are then convolved with a lowpass filter matched to the carrier frequency f_c, which introduces a second delay t_{LP}. The filtered I and Q components are then combined into a real valued voltage signal $v(t)$, which is then converted back into a digital stream of bits.

In the following chapters I will refer to a sample as the IQ-data of a sampled radio signal as seen by a receiver, meaning that channel effects are considered if not specified differently.

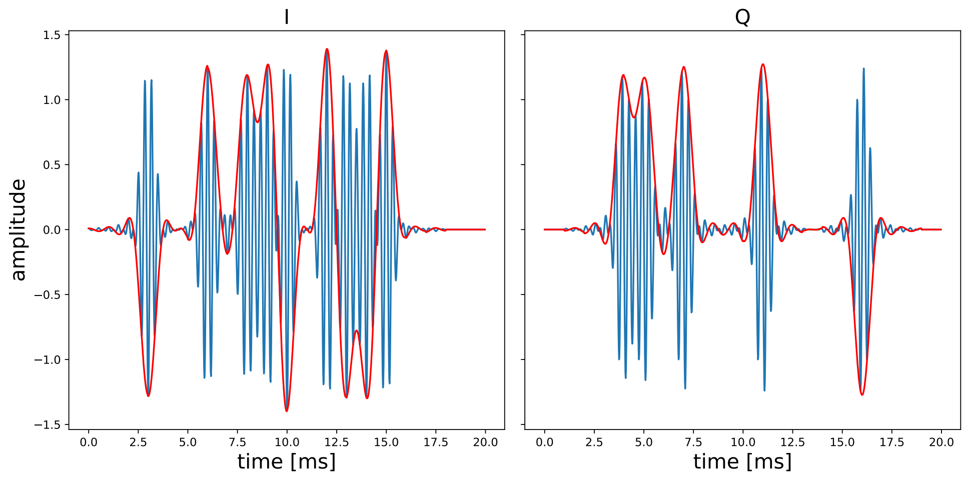


Figure 1.2: Example of I and Q representation of a radio signal modulated with QPSK. The blue lines represent the orthogonal basis functions of the carrier signal decomposition as described in 1.3. Their amplitude is modulated by the I and Q components shown in red.

2

Material and methods

This chapter outlines the dataset used for the modulation recognition task, the conversion of the samples into spike trains, the neuromorphic architecture and the computational framework of the proposed solution.

2.1 RadioML dataset

The source for the training and the test data is the RadioML dataset[14]. Specifically the 2016.10a version which includes sampled IQ-data for 11 modulations at 20 different signal to noise ratios (SNRs). Each class in the dataset contains 1000 samples and is indexed by a tuple `(mod, snr)`. SNR ranges from -20dB to +20dB at unit steps, whereas the list of modulations contains:

- 8 digital modulations: BPSK, QPSK, 8PSK, QAM16, QAM64, GFSK, CPFSK, PAM4
- 3 analog modulations: AM-SSB, AM-DSB, WBFM.

The samples are generated with the GNU radio[1] tool set using voice recordings for analog modulations and text from works of Shakespeare for digital modulations. Both source alphabets are randomized to guarantee equal symbol probabilities across all classes. Symbols are sampled from the alphabets and mapped to the respective baseband representation for each modulation using the `gr-mapper` module[2]. The IQ-data is obtained by sampling a finite impulse response (FIR) root-raised-cosine (RRC) filter at a specified rate. These samples are then transmitted through the channel simulation block, which models the following distorting effects:

- random processes for central frequency offset
- sample rate offset

[1]https://www.gnuradio.org/
[2]https://github.com/gr-vt/gr-mapper

- additive white Gaussian noise
- multi-path and selective fading.

The final samples are obtained by sampling the output of the channel simulation at a rate of 1 MHz for a duration of 128 µs. Each sample is then normalized to have unit energy and stored as 2x128 floating point array, where the rows correspond to the I and Q components.

Visual inspection of a subset of samples from the dataset (Figure 2.1) shows a certain degree of similarity between signals in a given class. Dissimilarity between signals from different classes can also be noticed in certain cases. However, notice the high level of similarity for signals belonging to the 3 PSK and the 2 QAM modulation classes. The reason for this can be understood by observing the constellation diagram for 8PSK, which contains the symbols of BPSK and QPSK (see Figure 1.1). At the same time, some of the symbols in the 8PSK constellation diagram are in the proximity of the 4 innermost symbols of QAM16. Since the underlying encoding protocol is the same and only the number of baseband symbols differs, the various distorting channel effects introduced in the communication system can increase the similarity of these symbols, which is noticeable in the IQ-data.

2.2 Conversion into spike trains

Information in the brain is transmitted over networks of neurons as sequences of spikes. It is processed by neurons and passed along synapses, which give rise to memory by strengthening the most relevant connections. Neuromorphic hardware processes information in a similar fashion, through networks of artificial silicon neurons and synapses that reproduce the dynamics of their real counterparts with bio-plausible time constants.

The IQ-data described in the previous section is thus not directly processable by a neuromorphic system, but must first be converted into spike trains. This is achieved with an algorithm that replicates the basic working principles of an Asynchronous Delta Modulator, as presented by Corradi et al[2]. In this pre-processing system an incoming signal V_{in} is converted into two spike trains, referred to as UP and DN channels. V_{in} is compared with two trailing thresholds V_{UP} and V_{DN}. When it crosses either one of them a spike is generated in the according channel and the threshold is updated to the current signal value plus or minus the threshold amplitude. The UP and DN channels therefore encode the value of the slope of the signal in the instantaneous firing rate of the spike trains, with positive values being stored in the UP channel and negative ones in the DN channel.

To emulate the self-timed clock-less nature of the analog circuit in [2] it is necessary to interpolate the IQ-data and resample it at a higher frequency.

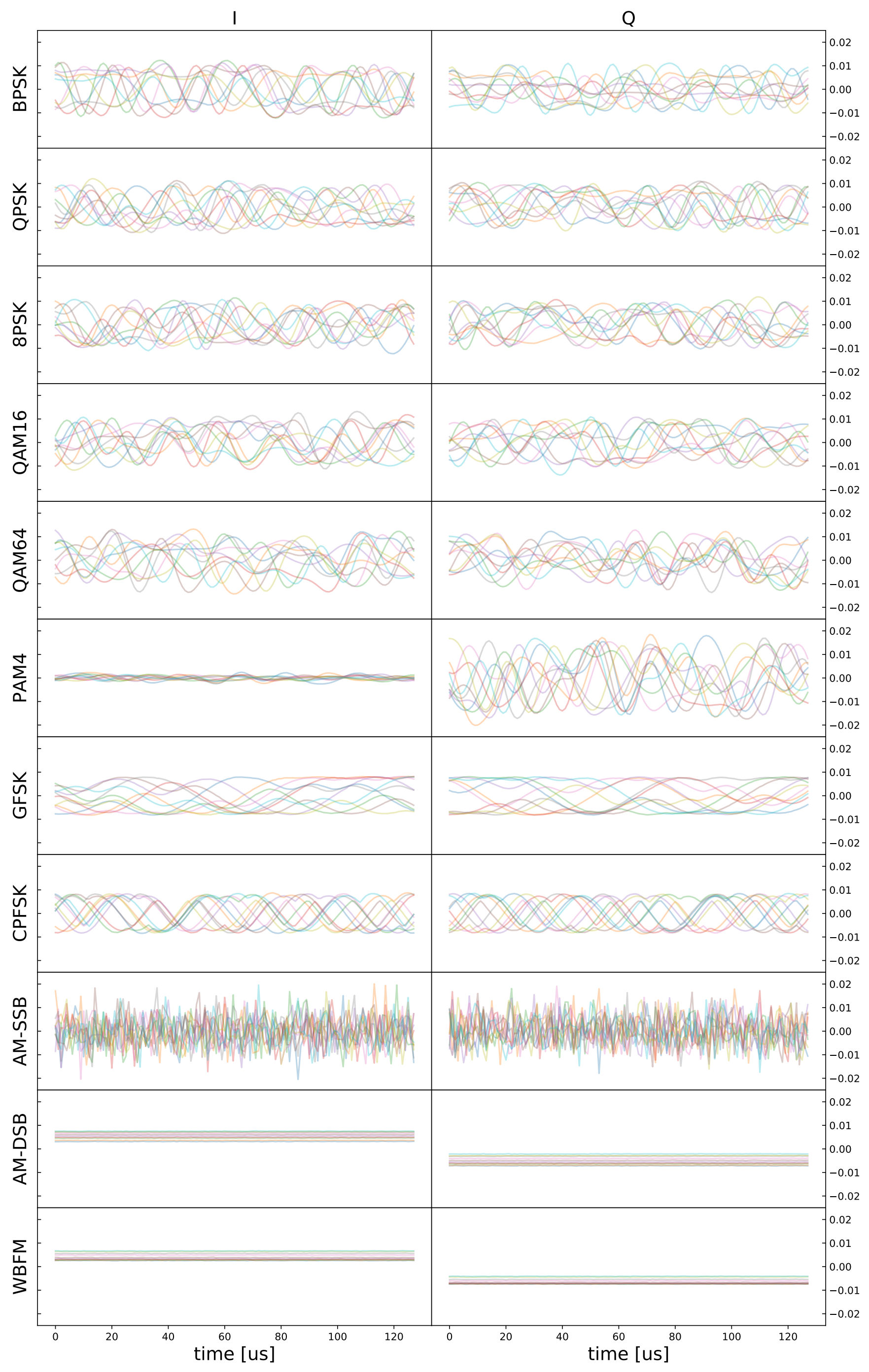


Figure 2.1: I and Q components of 10 samples at SNR of 18dB for each modulation class in the RadioML.2016.10a dataset.

This leads to a smaller discretization of time and a more precise spike timing. A further and more significant issue is however related to the time scale of the features in the radio signals. The samples in the RadioML dataset have a duration of 128 µs and are sampled at 1 MHz. Thus, any task-relevant spatio-temporal[3] pattern in the IQ-data has a time scale in the order of 10 µs − 100 µs. Neuromorphic hardware such as the ROLLS and the DYNAPs are designed to operate with bio-plausible signals, which present typical time scales of 100 ms, 3 order of magnitudes slower than the radio signals. To bypass this issue the time scale of the IQ-data must be stretched to the operational regime of the artificial neurons. The conversion of the I and Q components into spike trains adapted to neuromorphic hardware is thus carried out by the execution of the following algorithm:

Algorithm 1 Conversion of IQ-data into spike trains

```
Require: sample vector X, thresholds $V_{UP}$ and $V_{DN}$, interpolating function
    $f_{interp}$, resampling factor R and stretching factor S
 1: $T \leftarrow \text{Length}(X) \cdot R$
 2: $V_{in} \leftarrow f_{interp}$(T, X)
 3: $V_{dc} \leftarrow V_{in}[0]$
 4: indices                ▷ list of spike indices: 0 for UP and 1 for DN channel
 5: times                                                    ▷ list of spike times
 6: for $t \leftarrow 0, T$ do
 7:     if $(V_{dc} + V_{UP}) < V_{in}[t]$ then
 8:         indices ← 0
 9:         times ← $t$
10:         $V_{dc} \leftarrow V_{in}[t]$
11:     end if
12:     if $(V_{dc} - V_{DN}) > V_{in}[t]$ then
13:         indices ← 1
14:         times ← $t$
15:         $V_{dc} \leftarrow V_{in}[t]$
16:     end if
17: end for
18: return indices, $S \cdot$ times
```

The parameters of the algorithm are:

- V_{UP} and V_{DN}: UP and DN channel thresholds
- R: resampling factor, i.e. how many new IQ-data points are sampled in the interval of $1\mu s$

[3]The space coordinate refers either to the I or Q component. Therefore a spatio-temporal pattern in this context is a pattern of spikes across the UP and DN channels of the I and Q components.

- S: stretch factor, i.e. by how much the IQ-data time scale needs to be stretched
- f_{interp}: function used to interpolate between two data points in the IQ-data.

The parameters need to be optimized such that the reconstruction error is acceptable and the spike rate in the channels is sufficiently high to stimulate the artificial neurons. The first condition can be tested iteratively with a the following reconstruction algorithm:

Algorithm 2 Reconstruction of IQ-data from spike trains

```
Require: indices, times, $V_{UP}$ and $V_{DN}$
 1: $V_{rec}[0] \leftarrow 0$
 2: for t in times do
 3:     if indices[t]=0 then
 4:         $V_{rec}[t] \leftarrow V_{rec}[t] + V_{UP}$
 5:     end if
 6:     if indices[t]=1 then
 7:         $V_{rec}[t] \leftarrow V_{rec}[t] - V_{DN}$
 8:     end if
 9: end for
10: return $V_{rec}$
```

The latter condition is dependent of the biases of the neurons and synapses on the neuromorphic hardware, as well as the weights in the network. Optimization in this case is therefore more complicated and will be treated at a later point.

An example of spike trains generated through Algorithm 1 and of the reconstruction obtained with Algorithm 2 is shown in Figure 2.2 for a QPSK sample. The amplitude of the IQ-data is rescaled by a factor of 50 so that it ranges from -1 to +1. The reconstructed signals have been shifted so that the values at $t = 0$ coincide with the original signals to facilitate the comparison. The channel thresholds are set to 0.1 and the resampling factor is 200, thus increasing the original sampling rate of 1 MHz to 0.2 GHz. The original time interval of 128 µs is stretched out by a factor of 10^3 to 128 ms. These optimal values have been found by minimizing the reconstruction error

$$\varepsilon_{rec} = \frac{1}{T} \sum_{i=1}^{T} (V_{in} - V_{rec})^2 . \tag{2.1}$$

As shown in Figure 2.3, the reconstruction error is mostly affected by the threshold values V_{UP} and V_{DN}. The resampling factor does not seem to

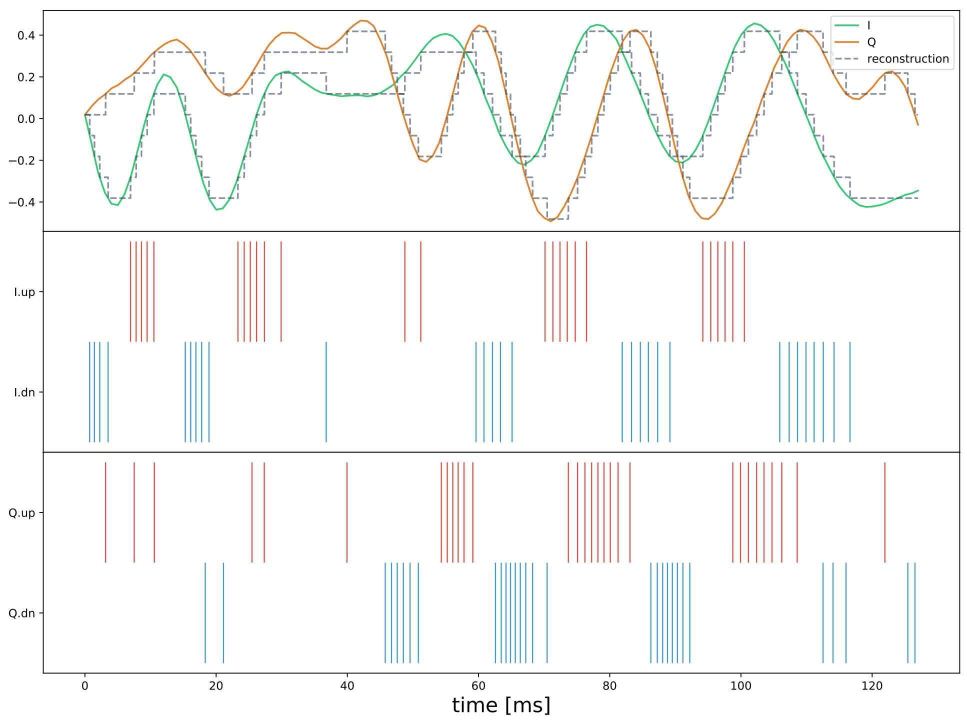


Figure 2.2: Conversion into spike trains and reconstruction of a sample with QPSK modulation. The I and Q components of the sample are shown in the top panel in green and orange, while the respective reconstruction obtained using Algorithm 2 is shown as dashed grey line. The two bottom panels show the spike trains generated through Algorithm 1 for each component. Red lines are spikes in the UP channel, whereas blue lines are spikes in the DN channel.

impact the conversion of signals into spikes and only needs to be larger than 1 for this example.

The results of Figure 2.4, which shows the reconstruction error for all the samples in the modulation classes at a SNR of 18dB, confirm that this choice of parameters leads to well behaved spike trains for most of the samples.

The higher reconstruction error in the AM-SSB modulation class is due to the higher frequency of the temporal features in the samples (see Figure 2.1). On the other hand, the reconstruction error is close to 0 for most of the samples in the AM-DSB and WBFM modulation classes because these samples have almost flat I and Q components (see Figure 2.1).

2.3 Neuromorphic architecture

The initial objective of this project was to develop the AMC system directly on the ROLLS neuromorphic processor. However, due to a failure of the

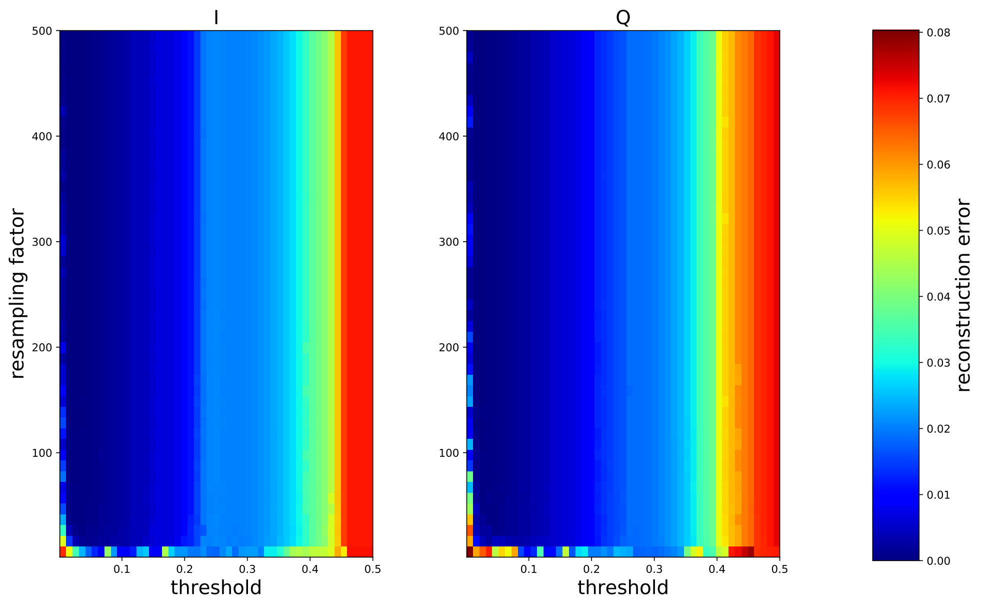


Figure 2.3: Reconstruction error for a sample with QPSK modulation as a function of the threshold amplitude for the UP and DN channels and the resampling factor. The plot shows a clear dependence between reconstruction error and threshold amplitude.

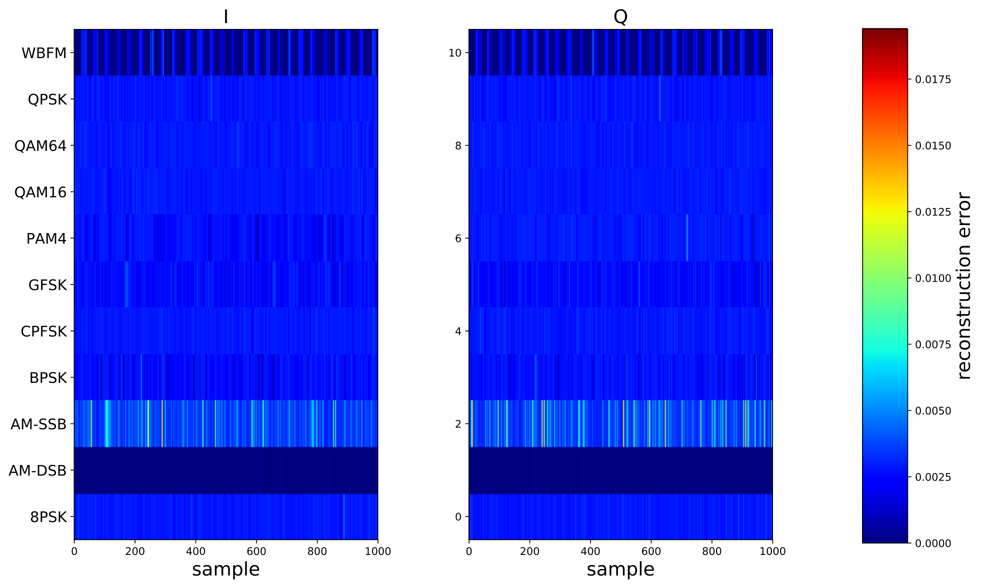


Figure 2.4: Reconstruction error for all samples in the modulation classes with SNR=18dB of the RadioML dataset. For most samples, the reconstruction is acceptable. Higher errors in the AM-SSB class is caused by higher frequency oscillations in the samples, whereas near zero error in AM-DSB and WBFM classes is due to almost flat I and Q components.

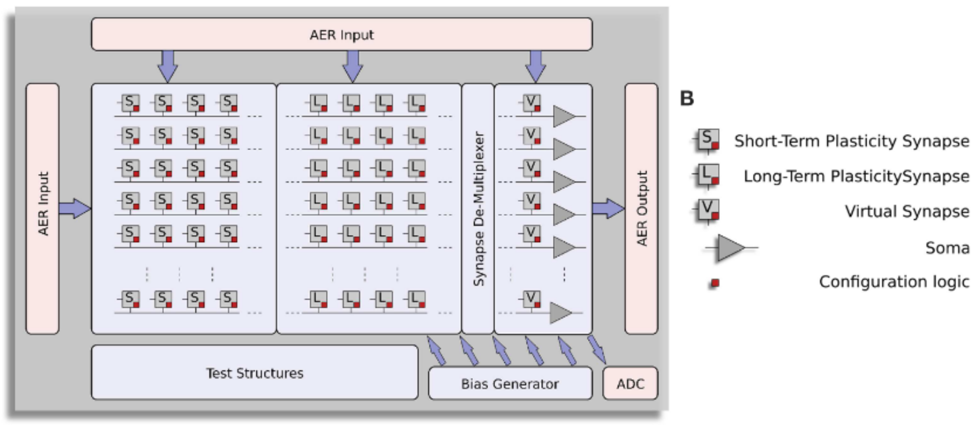


Figure 2.5: Architecture of the ROLLS neuromorphic processor[16].

chip I was not able to continue working on it and had to resort to simulations of the neural architecture using the Brian2[4] spiking neural network simulator and the Teili[5] toolbox, which among other things provides various neural and synaptic models, including those implemented in neuromorphic hardware. The combination of the two packages thus allows the simulation of spiking neural networks as if executed on neuromorphic hardware. The results from the simulations provide valuable insight into the various computational aspects related to the classification task at hand.

In the following sections I briefly describe the ROLLS neuromorphic processor before discussing the computational framework and the design of the neural network implemented in the simulations.

2.3.1 Reconfigurable neuromorphic processor

The Reconfigurable On-Line Learning Spiking (ROLLS) neuromorphic processor consists of 256 adaptive exponential integrate-and-fire neurons connected to 128k synapses[10][16]. The circuitry of the device is implemented with mixed signal analog/digital electronics and is fabricated using standard 180nm CMOS process, which yields a power consumption in typical experimental scenarios of approximately 4 mW. Two separate configurable arrays, each consisting of 64k synapses circuits, model short- and long-term plasticity mechanisms (see Figure 2.5). Spike-based Hebbian learning is implemented for the LTP synapses, while the STP synapses have configurable weights and can be excitatory or inhibitory.

A third array of linear integrator filter model ”virtual” excitatory and inhibitory synapses with shared synaptic weights and time constants. This

[4]https://brian2.readthedocs.io/en/stable/
[5]https://teili.readthedocs.io/en/latest/

group of synapses can be used to stimulate the neural network with input spikes from peripheral asynchronous I/O circuits that implement the Address-Event Representation (AER) protocol.

Configuration of the on-chip connectivity, synaptic and neural properties is made possible via digital blocks in each synapse and neuron circuit. This allows the user to design different network topologies, such as recurrent and deep neural networks, and to program different neural behaviors.

The Teili toolbox includes the DPI neuron model[1], which describes the dynamics of the neurons and synapses implemented on the ROLLS and DYNAPs neuromorphic processors.

2.3.2 Reservoir computing

The samples of the RadioML dataset consist of a sequence of baseband symbols modulated onto a carrier signal which has been transmitted through a simulated channel environment. The received signal is stored in complex baseband representation (I and Q components). In traditional radio communication systems, recovering information about the modulation used to generate a given radio signal involves the design of filters that match the transmitted representation of the baseband symbols for all possible modulations. As the incoming signal slides through the filters, peaks in the activation of the filters yield a basis for the classification task.

The neural network implemented must therefore be able to:

1. generate specific activation patterns for every baseband symbol
2. maintain a working memory of previous activation patterns.

Reservoir computing is a computational framework that has the potential to satisfy both requirements[12][21][11]. By randomly connecting N neurons via excitatory and inhibitory synapses, the reservoir can project inputs into a high-dimensional space, which is identified with the state-space of the network. A point in this space is defined by the current activation pattern of reservoir neurons. The random connectivity enables recurrent connections which can sustain an activation pattern and therefore enables a form of fading memory. Given an input spike pattern $u(t)$, the current state $x(t)$ of the reservoir is defined by

$$x(t) = f(W_{in}u(t) + W_{res}x(t - dt)), \tag{2.2}$$

where W_{in} are the weights of the input synapses, W_{res} are the weights of the recurrent synapses in the reservoir and f is the activation function of the neurons. Hence, upon presentation of an input pattern to the reservoir, the reservoir states at a later time should incorporate all the relevant temporal information of the input. The high-dimensionality of the state-space facilitates the linear separability of these states for different inputs and thus classification can be performed by adjusting the weights W_{out} of a linear readout layer given by

$$y(t) = W_{out} x(t), \tag{2.3}$$

where $y(t)$ is a vector that holds the current activation pattern of the output neurons, connected in all-to-all fashion to the reservoir neurons.

To summarize, a reservoir computing system is composed of three elements:

1. input layer: projects spike trains into the reservoir
2. reservoir layer: processes input and maps it to a high-dimensional state
3. readout layer: classifies high-dimensional state according to a ground truth.

I will describe each layer in greater detail in the following sections.

Input layer

Baseband symbols are encoded in the relative oscillations of the I and Q components. The delta modulator implemented in Algorithm 1 converts each time series into 2 channels (UP and DN) containing spike trains. The correlation in the IQ-data introduced by the modulation process is therefore reflected in the spike patterns of UP and DN channels of the different components.

The input spike trains should be projected into the subsequent layer with minimal loss of information and in a meaningful manner, depending on the connectivity model implemented in the reservoir.

The input layer implemented in the simulations consists of 8 spike generators which stimulate a layer of 4 input neurons via excitatory and inhibitory synapses of weight w_{gen}. The connectivity is shown in Figure 2.6. The input neurons then project their spikes to a subset of reservoir neurons via excitatory synapses with probability p_{IR}. The weight w_{inp} of the input synapses is fixed.

This configuration of the input layer allows to control the amount of spikes projected into the reservoir by tweaking the parameters of the input neurons and the weight of the generator synapses. Additionally, each input channel is represented in the spikes of the input neurons, with minimal loss of information, and can excite neurons in the reservoir.

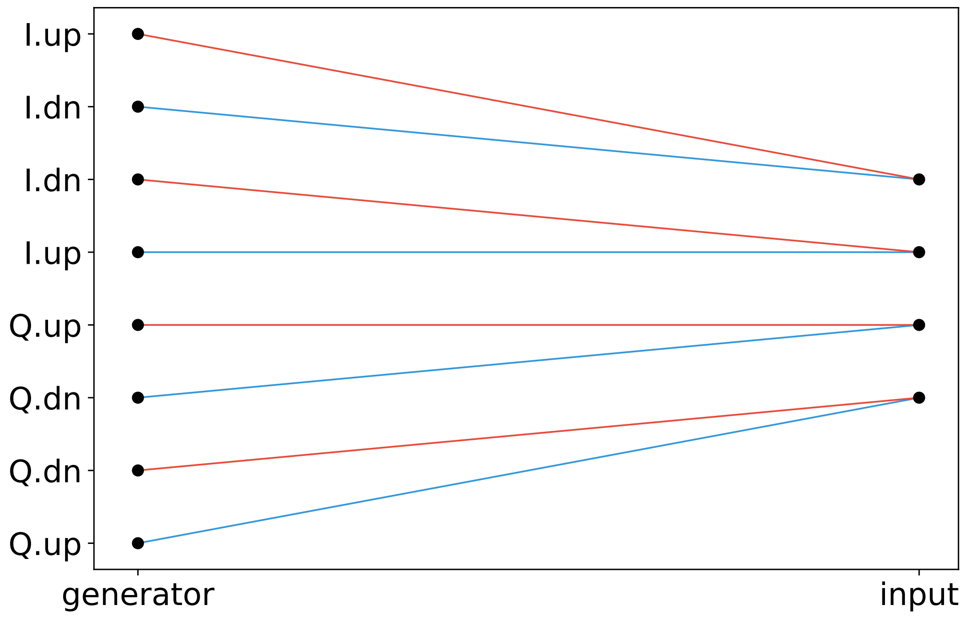


Figure 2.6: Connectivity between the spike generators and the 4 input neurons. Each neuron projects the input signal to a random subset of reservoir neurons. Excitatory and inhibitory synapses are represented by red and blue lines respectively.

Reservoir layer

The reservoir consists of N randomly connected excitatory and inhibitory neurons. The fraction f defines the amount of inhibitory neurons. To replicate mismatch between the electronic components of neurons and synapses in neuromorphic hardware, neural and synaptic time constants are sampled from Gaussian distributions. In a sufficiently large reservoir this heterogeneity is responsible for highly diversified recurrent loops, which are sensitive to different temporal patterns. Mismatch should therefore lead to more robust classification. Excitatory and inhibitory synaptic weights are also sampled from Gaussian distributions $\mathcal{N}_E(\mu_E, \sigma_E)$ and $\mathcal{N}_I(\mu_I, \sigma_I)$ respectively.

Regarding the recurrent connectivity, I implemented and tested two different models, as proposed in [19] and [8]. In both cases the connection probability between two neurons is dependant on the distance between them, as measured on the topology in which the reservoir neurons are embedded.

Schliebs connectivity

The reservoir neurons in this model are embedded in a 3-dimensional lattice of size $N_x \times N_y \times N_z$. A fraction f of the N neurons in the reservoir is inhibitory, while the rest are excitatory. The type of each neuron is chosen randomly and uniformly across the lattice. The distance dependent probability of connection for two neurons A and B is defined as

$$p(A,B) \propto \Gamma(A,B) \exp\left(-\frac{d(A,B)^2}{2\lambda^2}\right), \tag{2.4}$$

where the distance function $d(A,B)$ is the Euclidean distance as measured on the lattice and λ controls the density of connection. The amplitude Γ of the probability function depends on the type of the neurons and is defined as

$$\Gamma(A,B) = \begin{cases} \gamma_{inh-inh}, & \text{if A and B are inhibitory} \\ \gamma_{inh-exc}, & \text{if A is inhibitory and B is excitatory} \\ \gamma_{exc-inh}, & \text{if A is excitatory and B is inhibitory} \\ \gamma_{exc-exc}, & \text{if A and B are excitatory.} \end{cases} \tag{2.5}$$

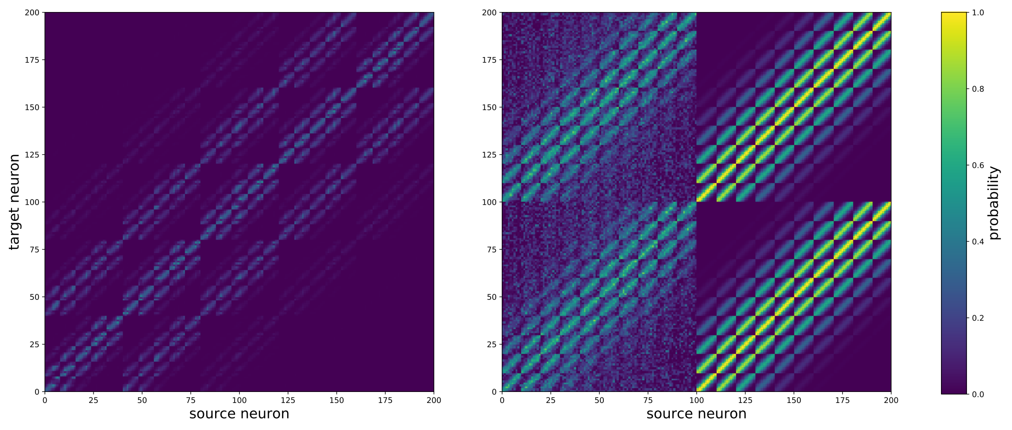
By setting the amplitudes of Γ, one can selectively regulate the range of connections depending on the neuron type.

Hennequin connectivity

The reservoir neurons in this model are organized in two 2-dimensional grids of size $N_x \times N_y$ according to their type. Excitatory neurons can make local and long-range connections to other neurons, while connections from inhibitory neurons are strictly local. This type of connectivity has been found in experimental data and theoretical models of the neocortex[22]. The probability of connection between two neurons A and B is thus defined as

$$\begin{aligned} p(A,B) \propto p_A^{local} \exp\left(-\frac{d(A,B)^2}{2\lambda^2}\right) + \\ (1 - p_A^{local})\frac{1}{K}\sum_{i=1}^{K} \exp\left(-\frac{d(A,L_i)^2}{2\lambda^2}\right), \end{aligned} \tag{2.6}$$

where $d(A,B)$ is the Euclidean distance as measured on the grid, p_A^{local} describes the fraction of local connections made by neuron A, K is the number of long-range connections and $\{L_i | i = 1, \ldots, K\}$ is a set of target neurons for neuron A drawn randomly and uniformly on the grid. Patchy connectivity as described above is obtained by setting $p_A^{local} = 1.0$ for all inhibitory neurons and $p_A^{local} < 1.0$ for all excitatory neurons.



(a) Schliebs (b) Hennequin

Figure 2.7: Connection probability matrices for two different connectivity models of a reservoir with $N = 200$ neurons.

Readout layer

A supervised learning process can be implemented on-chip using a spike-based Hebbian learning rule together with teacher signals that stimulate the neurons in the output layer according to the ground truth label.

The DPI synapse model implements the following rule:

$$\begin{cases} V_w = V_w + \Delta w & \text{if } V_{mem} > V_{thm} \\ & \text{and } V_{thk1} < V_{Ca} < V_{thk3} \\ V_w = V_w - \Delta w & \text{if } V_{mem} < V_{thm} \\ & \text{and } V_{thk1} < V_{Ca} < V_{thk2}, \end{cases} \tag{2.7}$$

in which V_{mem} represents the post-synaptic membrane potential at the time of pre-synaptic spike and V_{Ca} describes the calcium concentration of the neuron, which is proportional to its recent firing activity. In this case the weights of the readout synapses are updated according to the relative timing of pre- and post-synaptic spikes. The learning behavior can be adjusted by tuning the values of the thresholds V_{thm}, V_{thk1}, V_{thk2} and V_{thk3}.

For the purpose of testing the classification capabilities of the reservoir, the learning process is replaced by an offline linear classifier. The activity of the reservoir for each input sample is recorded and binned into k time slots of duration Δt to obtain state vectors of the reservoir. A logistic regression classifier is then trained at every time slot k on the state vectors collected at that time from all samples.

2.4 Parameter tuning

Proper tuning of the model parameters summarized in Table 2.1 is crucial for the correct and stable behavior of the reservoir.

The parameter optimization process starts with adapting the dynamics of the neurons and synapses to the time scales of the information that needs to be extracted from the input signals. In devices and simulations that use the DPI neuron and synapse model, this involves adjusting the currents associated with the various properties. The neuron time constant, for example, is defined by the following equation:

$$\tau = \frac{C_{mem} U_T}{\kappa I_\tau} \tag{2.8}$$

where C_{mem} is the DPI neuron membrane capacitance, U_T is the thermal voltage and I_τ is a current. Given the characteristics of the temporal features in the RadioML dataset, a mean time constant of 20 ms for the reservoir neurons is appropriate, whereas for the input neurons a smaller mean time constant of 5 ms was chosen in order to filter out high firing rates in some of the input samples. The spiking behavior of the input and reservoir neurons can be further controlled by adjusting the spiking threshold current I_{spkthr}, which was set to 0.2 nA for both groups. The mean time constant of the synapses in the reservoir was set to 7 ms. Device mismatch was simulated by setting the standard deviation at a fraction $\eta = 0.2$ of the parameter value.

In a subsequent step, all the parameters related to the connectivity of the neurons were optimized. The weight w_{gen} was adjusted such that the spike patterns of the input layer closely track those of the spike generators. The weight of the synapses to the reservoir had to be optimized together with those of the reservoir in order to maintain the dynamics in a critical regime of recurrent activity, but not chaotic. Figures 2.8 and 2.9 illustrate an iteration in the process of tuning the parameters using the sample of Figure 2.2 as an input signal.

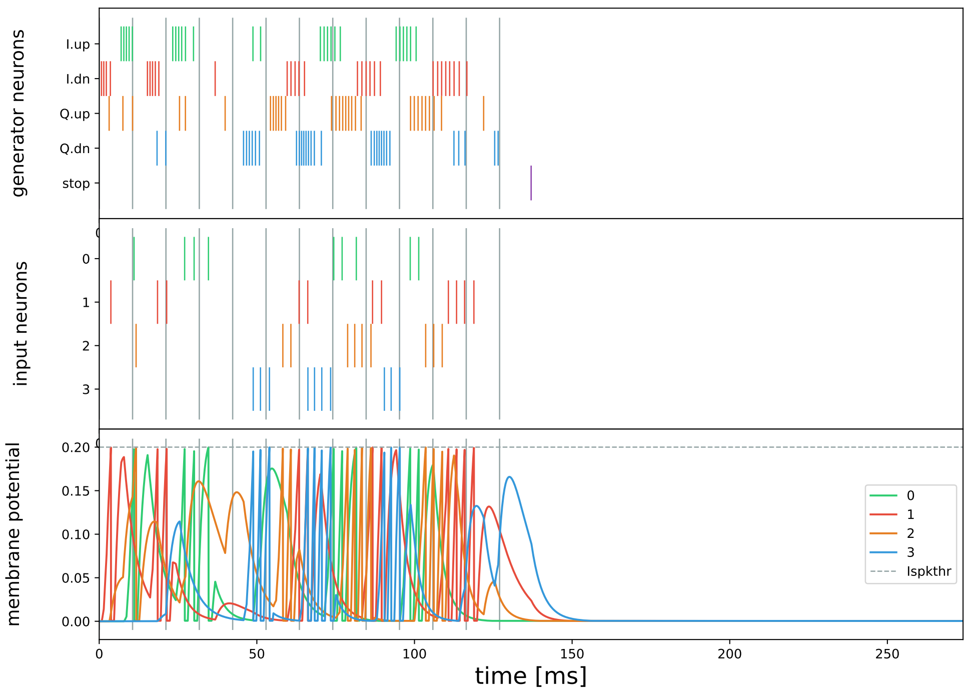


Figure 2.8: Input spike trains from the generators (top) and input neurons (middle). The membrane potential of the input neurons is shown in the bottom panel. Vertical lines define 10 ms intervals that roughly coincide with half period in the input signal oscillation.

connectivity	parameter	description
	N	number of reservoir neurons
	N_{inp}	number of input neurons
	f	fraction of inhibitory neurons
Schliebs	N_x	grid size along the x-axis
	N_y	grid size along the y-axis
	N_z	grid size along the z-axis
	$\gamma_{inh-inh}$	connection probability amplitude between inhibitory neurons
	$\gamma_{inh-exh}$	connection probability amplitude from inhibitory to excitatory neurons
	$\gamma_{exc-inh}$	connection probability amplitude from excitatory to inhibitory neurons
	$\gamma_{exc-exc}$	connection probability amplitude between excitatory neurons
Hennequin	N_x	grid size along the x-axis
	N_y	grid size along the y-axis
	p_{exc}^{local}	fraction of local connections for excitatory neurons
	p_{inh}^{local}	fraction of local connections for inhibitory neurons
	k	number of long-range connections
	λ	density of connection
	w_{gen}	weight of synapses from spike generators to input neurons
	w_{inp}	weight of synpases from input neurons or generators to reservoir neurons
	μ_{res}^{exc}	mean of weight distribution of excitatory reservoir synapses
	σ_{res}^{exc}	standard deviation of weight distribution of excitatory reservoir synapses
	μ_{res}^{inh}	mean of weight distribution of inhibitory reservoir synapses
	σ_{res}^{inh}	standard deviation of weight distribution of inhibitory reservoir synapses
	μ_{τ}	mean of neural time constant distribution
	μ_{τ_s}	mean of synaptic time constant distribution
	η	mismatch percentage

Table 2.1: List of parameters of the neural network implemented for the automatic modulation classification task.

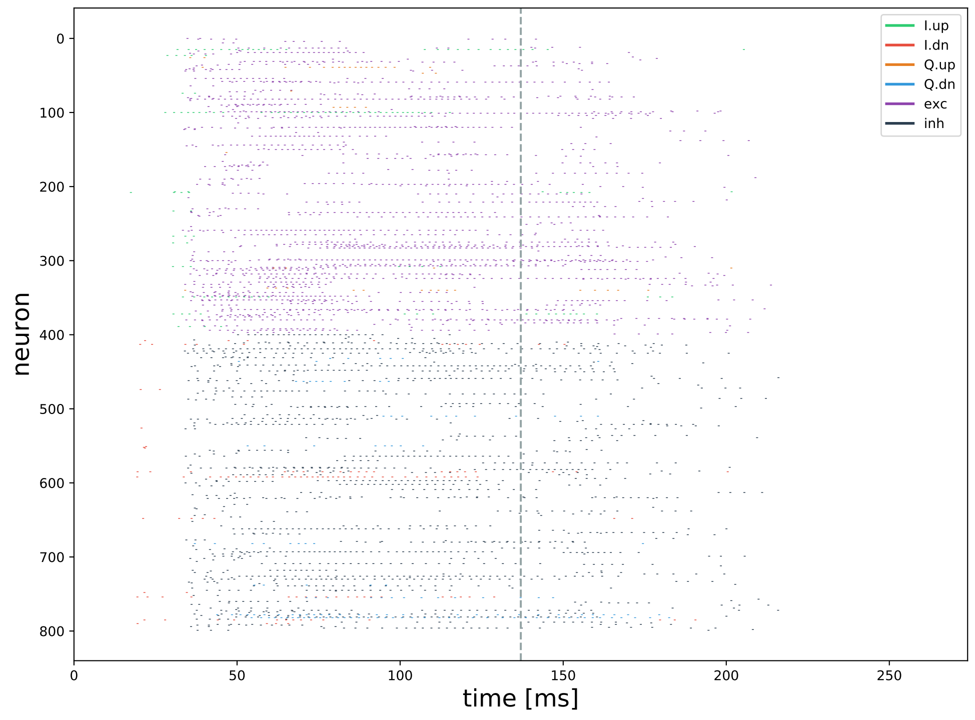


Figure 2.9: Activity of a reservoir with Hennequin connectivity during stimulation with an input sample with QPSK modulation. The dashed vertical line represents the end of the input signal.

3

Results

In this chapter I discuss the issues encountered and the results obtained from simulations of the neuromorphic modulation recognition system.

3.1 Issues encountered

Over the course of this master thesis, the most challenging part has always been the proper tuning of the many parameters of the neuromorphic architecture, both in hardware as well as in simulations. Often a set of neural and synaptic parameters would lead to separable activation patterns for a small set of samples of some modulation class, but fail to generalize to other classes. The nature of the particular classification task at hand thus requires the processing of a fair amount of samples in order to asses the performance of the system for a particular set of parameters. Devising methods that accelerate this process would be beneficial to the adoption of neuromorphic processors in different application domains.

Regarding the models of connectivity, I found it harder to tune the parameters of the Schliebs model, where excitatory and inhibitory neurons are randomly and uniformly distributed in a 3-dimensional cube and connected with a distance dependant probability. The reason for this might be related to properties of the connectivity matrix, which influence the balance between excitation and inhibition, but also the way recurrent activity propagates within the network and the overall stability of the system. A failed attempt at tuning the reservoir with Schliebs connectivity is shown in Figure 3.1. The same sample as in the previous figures was used as input stimulus.

3.1.1 E-I balance

The balance between excitatory and inhibitory synaptic weights has been observed in experimental data of cortical circuits[20][23]. Together with

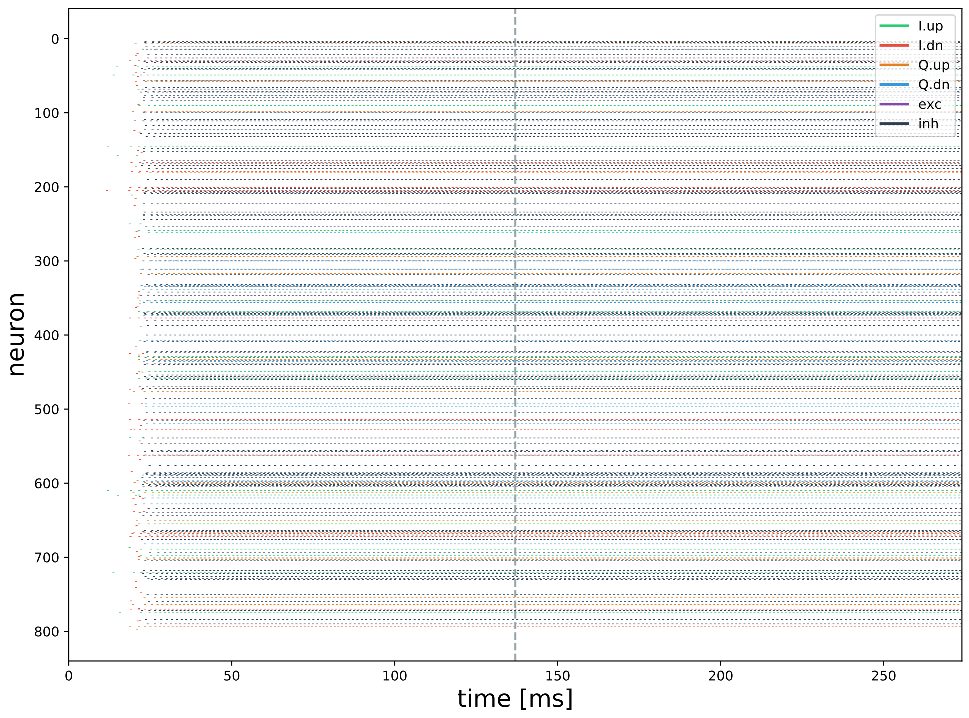
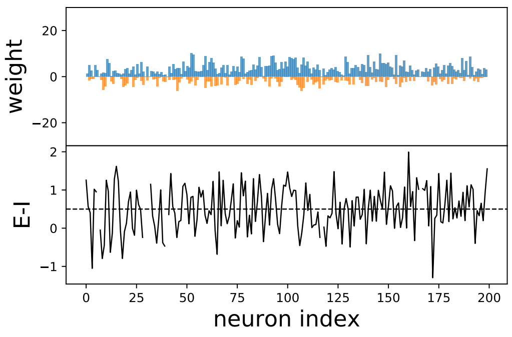
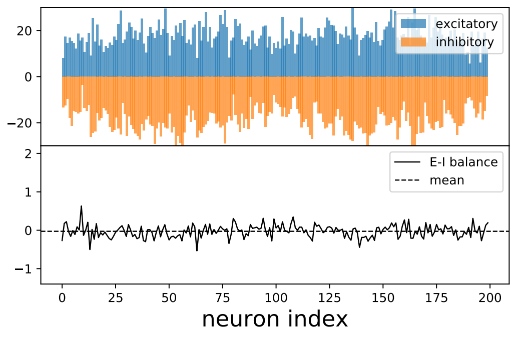


Figure 3.1: Unstable reservoir dynamics during tuning of the network parameters of the reservoir with Schliebs connectivity. The dashed vertical line represents the end of the input signal.

synaptic plasticity, it is speculated that E-I balance plays an important role in controlling the stability of the dynamics during stimulation with input signals by adjusting the weights of the synapses at different time scales[9]. In reservoir computing however, the weights are fixed at network initialization, leaving the system with no mechanism for further adaptation to input signals. It is thus important to properly scale the excitatory and inhibitory weights of the reservoir synapses. Figure 3.2 shows the balance of excitatory and inhibitory synaptic input weights for all neurons in the reservoir in the two models of connectivity described in Section 2.3.2. The weights were scaled to be withing the range $[-1, 1]$ to facilitate the comparison. Even though the Schliebs connectivity model has fewer connections overall, the excitatory and inhibitory weights are less balanced compared to those of the Hennequin model. The latter model is by design more prone to satisfy E-I balance, since excitatory neurons are connected to local inhibitory neurons, whereas in the former model inhibition occurs over few long range connections. This could explain the difficulty in finding proper parameters that lead to stable dynamics of the reservoir with Schliebs connectivity.



(a) Schliebs



(b) Hennequin

Figure 3.2: Excitatory and inhibitory balance of synaptic input weights for each neuron in a reservoirs of size $N = 200$ with different connectivity models. The black dashed line represents the mean EI balance of the network, which is lower for the reservoir with Hennequin connectivity.

3.2 Classification of PSK modulations

The following initial results were obtained using a reservoir of $N = 800$ neurons with Hennequin connectivity. The classification task was restricted to 3 digital modulations classes (8BPSK, QPSK, 8PSK) and 50 samples per class. Each sample had a maximal input duration of 128 ms. Recurrent activity was observed in a time window of 500 ms after stimulation finished (see Figure 3.3). The membrane potential and the synaptic currents were resetted to their resting values after a readout period of 628 ms. The total stimulation time amounted to 94.2 s, which required about 1 h to simulate.

State vectors for each sample were obtained by counting the number of spikes in a time window of 5 ms for every neuron over the duration of the readout time. This resulted in a total of 125 state vectors for each sample.

The time evolution of the classification accuracy is shown in Figure 3.4 for classifiers trained on the whole dataset. The accuracy increases gradually at the beginning and reaches its maximum after 120 ms from the onset of stimulation, which coincides approximately with the total duration of the input signal. The reservoir thus does accumulate memory about the input signals and is able to project them in separable regions of the high-dimensional state space. Accuracy drops gradually after the end of stimulation, as expected.

To see how this extends to never before seen signals I measure the accuracy on a small test dataset. The results in Figure 3.5 show peaks in accuracy of 46% at $t = 65$ ms and $t = 235$ ms. The average accuracy is 35%, which is only slightly above the accuracy of a random classifier, indicated by the red dashed line. Below-chance performance is observed at several time steps, particularly at the beginning of stimulation and at later times. This is likely

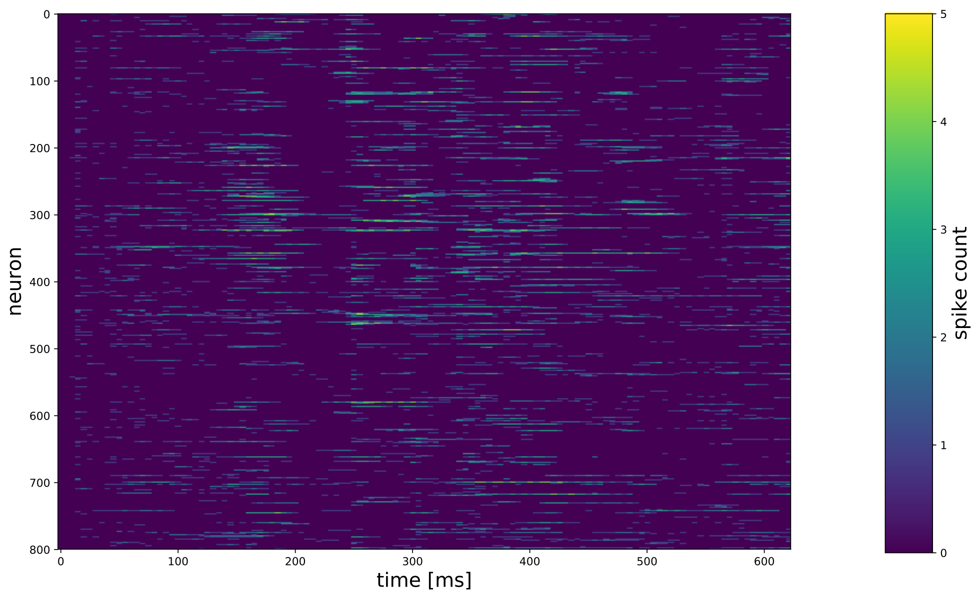


Figure 3.3: Example of a reservoir readout for an input signal with QPSK modulation. Sustained recurrent activity can be observed after the end of stimulation at $t = 128\,\text{ms}$.

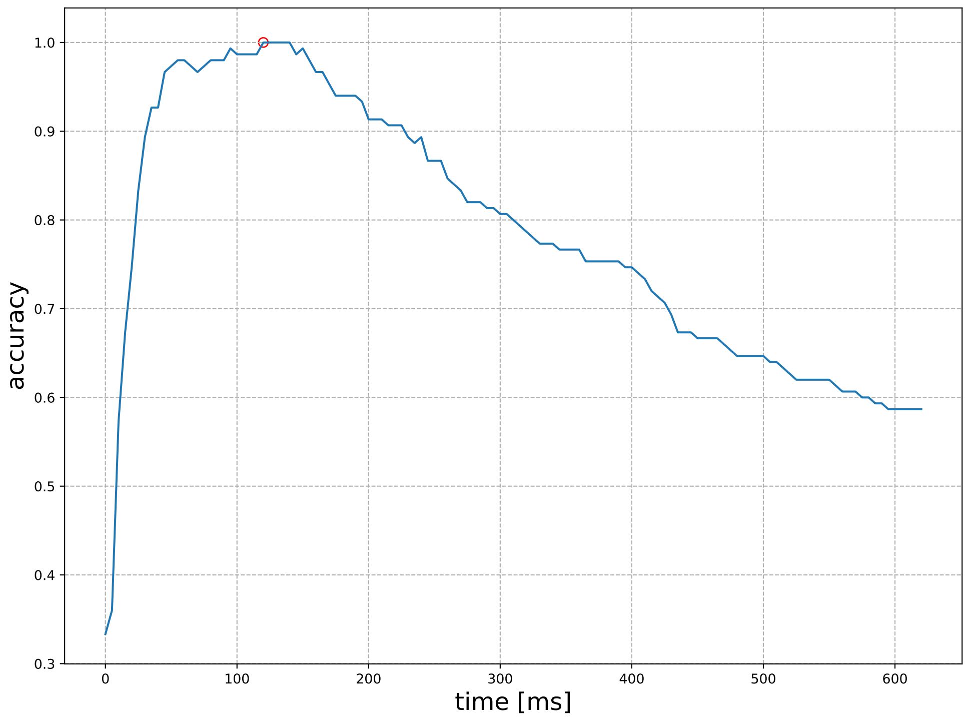


Figure 3.4: Classification accuracy measured on the training dataset as a function of time for the classification of 3 digital modulations (BPSK, QPSK, 8PSK) using a reservoir of $N = 800$ neurons with Hennequin connectivity.

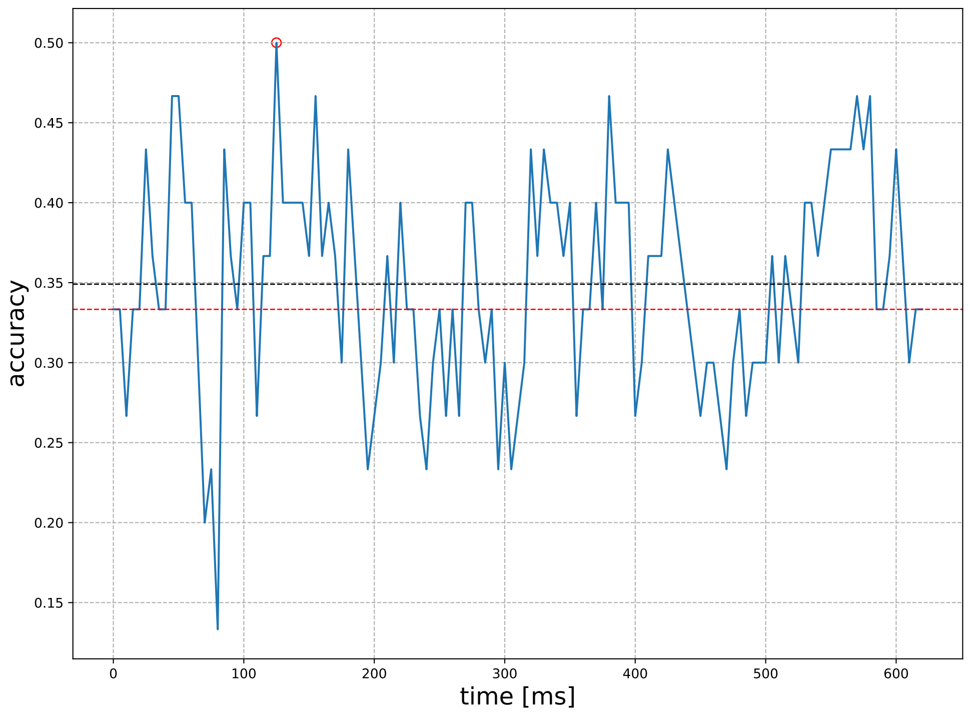


Figure 3.5: Classification accuracy measured on a test dataset as a function of time for the classification of 3 digital modulations (BPSK, QPSK, 8PSK) using a reservoir of $N = 800$ neurons with Hennequin connectivity.

due to a lack of generalization caused by the relatively small size of the dataset.

To understand the input selectivity of the neurons in the reservoir I compute the feature importance i_n for each neuron, defined as

$$i_n = \sigma_n \cdot \gamma_n \tag{3.1}$$

where σ_n is the standard deviation of the feature values over the whole dataset and γ_n is the n-th coefficient of the linear regression model. In this specific case σ_n is obtained from the distribution of spike counts for neuron n over the set of state vectors at time t. The feature importance of excitatory and inhibitory neurons is shown in Figure 3.6 for all 3 modulation classes.

Out of 800 neurons, about $\approx 20\%$ are utilized by the classifier to distinguish between the 3 modulations. Figure 3.7 shows the states extracted from the reservoir at $t = 120\,\text{ms}$. Neurons with feature importance $|i_n| > 0.05$ are indicated by ticks in the top axis. Although the state vectors look quite similar at a first glance, it is important to remember that the whole purpose of projecting the input signal into the reservoir is to exploit the high-dimensional

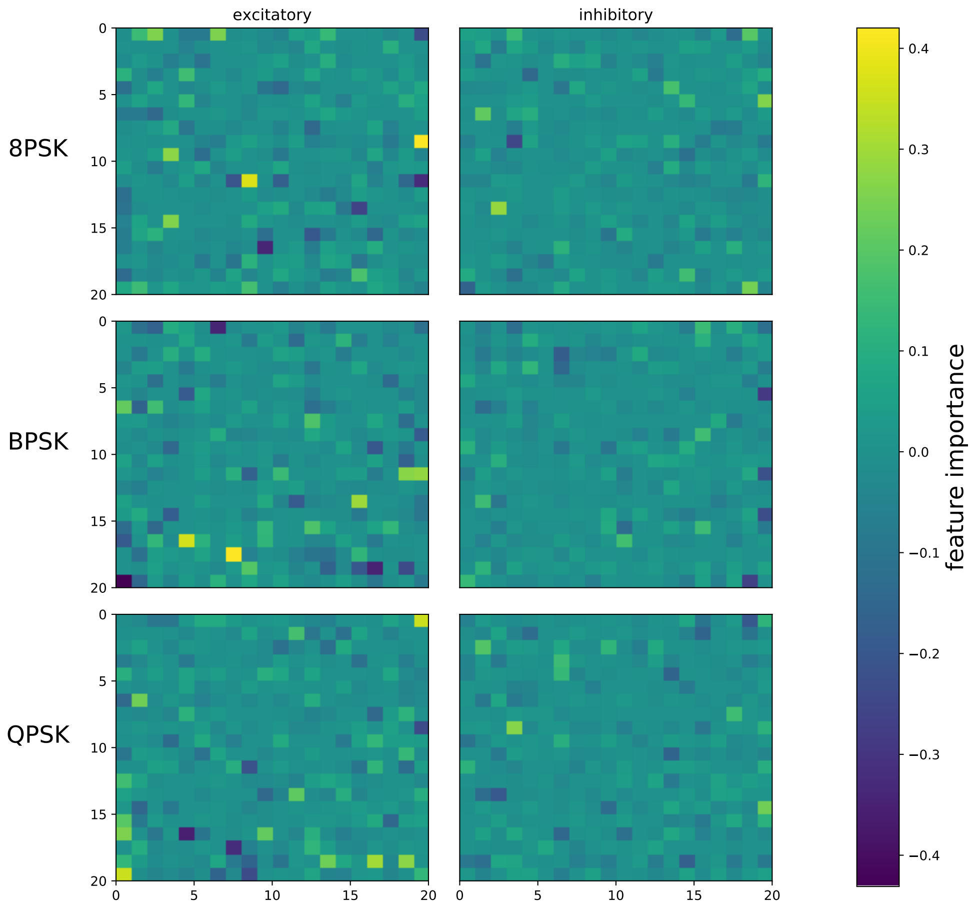


Figure 3.6: Feature importance of excitatory and inhibitory neurons (as defined by Eq. 3.1) at time of maximum accuracy $t = 120\,\text{ms}$ for the classification of 3 digital modulations (BPSK, QPSK, 8PSK) using a reservoir of $N = 800$ neurons with Hennequin connectivity.

representation produced by activation pattern of the neurons. To classify the input signals, only a few dimensions with enough variability are necessary.

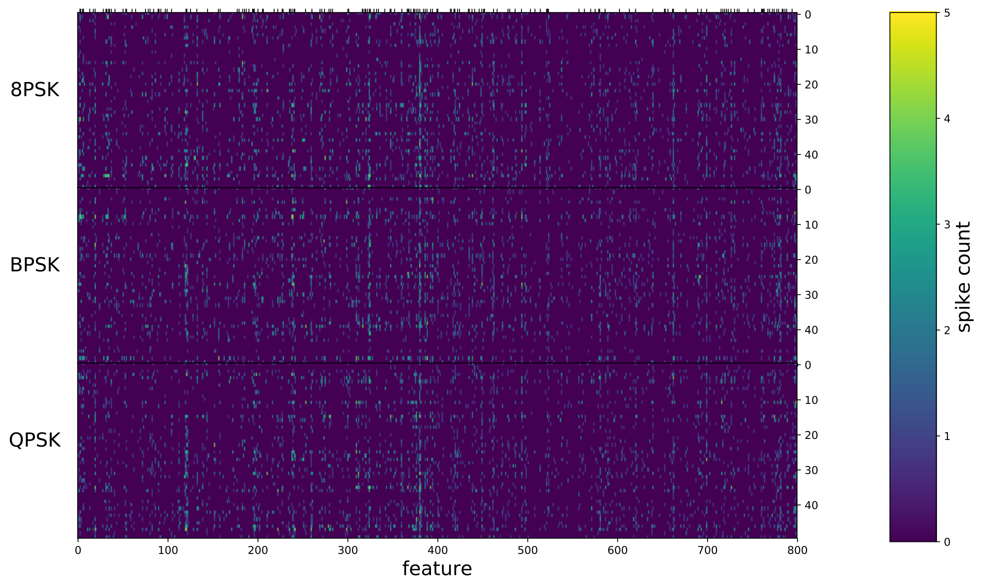


Figure 3.7: State vectors of all samples at time of maximum accuracy $t = 120\,\mathrm{ms}$ for the classification of 3 digital modulations (BPSK, QPSK, 8PSK) using a reservoir of $N = 800$ neurons with Hennequin connectivity. Neurons with an absolute feature importance larger than 0.05 are indicated by ticks on the top x-axis.

4

Conclusions

The methods and results of this work demonstrate the feasibility of classification of radio signal modulations using reservoir computing on neuromorphic processors. More trials need to be performed and the classification task should be extended to more classes, possibly involving a larger number of samples.

With regards to the neural architecture itself, various improvements should be considered in future work. Most importantly the implementation of plasticity mechanism to regulate the activity at various time scales. Short-term plasticity could be used to control the activity of the reservoir in an unsupervised manner, whereas long-term plasticity could help increase the separability of the state vectors, thus facilitating classification. Deep architectures, as proposed by Gallicchio et al.[6], could also be adopted in order to facilitate learning at different time scales.

Acknowledgements

I am thankful to Prof. Giacomo Indiveri and Prof. Tobi Delbruck for giving me the opportunity to work on this project. It has been an intense but stimulating learning experience. I would also like to extend my gratitude to Raphaela Kreiser and Alpha Renner, who supported me throughout the last 6 months and kindly offered their time and expertise during the most challenging parts of the project. Ultimately, I am thankful to all the wonderful people at the Institute of Neuroinformatics for the memorable time spent together.

Bibliography

[1] Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi, and Giacomo Indiveri. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of the IEEE*, 102(9):1367–1388, 2014.

[2] Federico Corradi and Giacomo Indiveri. A neuromorphic event-based neural recording system for smart brain-machine-interfaces. *IEEE transactions on biomedical circuits and systems*, 9(5):699–709, 2015.

[3] Joseph A Downey, Dale J Mortensen, Michael A Evans, Janette C Briones, and Nicholas Tollis. Adaptive coding and modulation experiment with nasa's space communication and navigation testbed. 2016.

[4] Joseph A Downey, Dale J Mortensen, Michael A Evans, and Nicholas S Tollis. Variable coding and modulation experiment using nasa's space communication and navigation testbed. 2016.

[5] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen. Multi-objective reinforcement learning-based deep neural networks for cognitive space communications. In *2017 Cognitive Communications for Aerospace Applications Workshop (CCAA)*, pages 1–8, June 2017.

[6] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[7] T. M. Hackett, S. G. Bilén, P. V. R. Ferreira, A. M. Wyglinski, and R. C. Reinhart. Implementation of a space communications cognitive engine. In *2017 Cognitive Communications for Aerospace Applications Workshop (CCAA)*, pages 1–7, June 2017.

[8] Guillaume Hennequin. *Stability and amplification in plastic cortical circuits.* dissertation, EPFL, 2013.

[9] Guillaume Hennequin, Everton J Agnes, and Tim P Vogels. Inhibitory plasticity: balance, control, and codependence. *Annual Review of Neuroscience*, 40:557–579, 2017.

[10] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André Van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, et al. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73, 2011.

[11] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[12] Wolfgang Maass, Prashant Joshi, and Eduardo D Sontag. Computational aspects of feedback in neural circuits. *PLoS computational biology*, 3(1):e165, 2007.

[13] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE transactions on biomedical circuits and systems*, 12(1):106–122, 2018.

[14] Timothy J O'shea and Nathan West. Radio machine learning dataset generation with gnu radio. In *Proceedings of the GNU Radio Conference*, volume 1, 2016.

[15] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *International conference on engineering applications of neural networks*, pages 213–226. Springer, 2016.

[16] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141, 2015.

[17] Radhika Radhakrishnan, William W Edmonson, Fatemeh Afghah, Ramon Martinez Rodriguez-Osorio, Frank Pinto, and Scott C Burleigh. Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view. *IEEE Communications Surveys & Tutorials*, 18(4):2442–2473, 2016.

[18] Richard Reinhart and James P Lux. Space-based reconfigurable software defined radio test bed aboard international space station. In *SpaceOps 2014 conference*, page 1612, 2014.

[19] Stefan Schliebs, Haza Nuzly Abdull Hamed, and Nikola Kasabov. Reservoir-based evolving spiking neural network for spatio-temporal pattern recognition. In *International Conference on Neural Information Processing*, pages 160–168. Springer, 2011.

[20] Yousheng Shu, Andrea Hasenstaub, and David A McCormick. Turning on and off recurrent balanced cortical activity. *Nature*, 423(6937):288, 2003.

[21] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: a review. *Neural Networks*, 2019.

[22] Nicole Voges, Almut Schüz, Ad Aertsen, and Stefan Rotter. A modeler's view on the spatial structure of intrinsic horizontal connectivity in the neocortex. *Progress in neurobiology*, 92(3):277–292, 2010.

[23] Shanglin Zhou and Yuguo Yu. Synaptic ei balance underlies efficient neural coding. *Frontiers in Neuroscience*, 12:46, 2018.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

CLASSIFICATION OF RADIO SIGNALS ON A NEUROMORPHIC PROCESSOR IN SPACE

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
BORTONE	MASSIMO

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZÜRICH, 08/04/2019	Massimo Bortone

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.