TUM School of Computation, Information and Technology
Technische Universität München

Radar Signal Processing with Spiking Neural Networks and Resonate-and-Fire Encoding

Julian Bernhard Christian Hille

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Stefan Leutenegger

Prüfende der Dissertation:

1. Prof. Dr.-Ing. habil. Alois Christian Knoll
2. Prof. Dr.-Ing. habil. Erwin Biebl

Die Dissertation wurde am 22.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 25.04.2024 angenommen.

Abstract

Spiking Neural Networks (SNNs) are the third generation of Artificial Neural Networks (ANNs) that mimic the biological dynamics observed in neuroscience. These networks use discrete electrical impulses, or spikes, to represent, transport, and process information. The encoding of the input signal into a spike train is a critical step in enabling efficient SNNs for signal processing applications.

Nature has perfected the low-power processing of data and signals throughout thousands of years. Inspired by this efficient way of processing, researchers have developed abstractions and simplifications in the form of artificial neurons. SNNs resemble this biological inspiration more closely. They have shown several advantages over traditional ANNs, including reduced computational complexity, smaller architectures, higher noise robustness, and improved energy efficiency.

The main contribution of this thesis is the proposal of a data stream encoding method based on Resonate-and-Fire (R&F) neurons. These neurons transform sequential input streams into spatio-temporal spike trains, where each spike carries information about the individual frequencies, amplitudes, and phases that form the signal. The spatial-dimensional spike trains improve the classification performance of the subsequent SNN population.

For the validation of the proposed encoding, we conducted a series of selected signal processing applications like radar interference detection, phase detection, and angle-of-arrival estimation. The results demonstrate that the R&F encoding outperforms existing encoding methods in terms of efficiency and the accuracy of the subsequent SNN population. The proposed SNN architectures show comparable performance to state-of-the-art solutions while providing a stream processing capability, faster classification, noise suppression ability, and sparse spike events.

The developed architectures provide evidence that SNNs with R&F encoding are viable alternatives to traditional methods and contribute to understanding the information processing within spiking neurons. The capability to directly process sensor signals enables highly efficient hardware-accelerated neuromorphic signal processing in various energy-constrained systems.

Zusammenfassung

Pulsende Neuronale Netze (SNN) sind die dritte Generation künstlicher neuronaler Netze (ANNs), die neurowissenschaftliche Erkenntnisse in deren Dynamik abbilden. Diese Netzwerke nutzen diskrete elektrische Impulse, sogenannte Spikes, um Informationen darzustellen, zu transportieren und zu verarbeiten. Die Kodierung der Eingangssignale in Pulsfolgen ist ein entscheidender Schritt, um effiziente SNNs für Signalverarbeitungsanwendungen zu ermöglichen.

Die Natur hat über Jahrtausende die energiesparende Verarbeitung von Daten und Signalen perfektioniert. Inspiriert von dieser effizienten Art der Verarbeitung haben Forscher Abstraktionen und Vereinfachungen in Form künstlicher Neuronen entwickelt. SNNs sind dem biologischen Vorbild am ähnlichsten und modellieren eine Vielzahl von Beobachtungen aus der Neurowissenschaft. Sie haben gegenüber den traditionellen ANNs mehrere Vorteile, darunter eine geringere Rechenkomplexität, kleinere Architekturen, eine höhere Robustheit gegenüber Rauschen und eine verbesserte Energieeffizienz.

Der wichtigste Beitrag dieser Arbeit ist eine Methode zur Kodierung von Datenströmen in Pulsfolgen auf der Grundlage von resonierenden Neuronen. Diese Resonate-and-Fire (R&F) Neuronen wandeln sequenzielle Eingabeströme in räumlich-zeitliche Pulsfolgen um, wobei jeder Spike Informationen über die einzelnen Frequenzen, Amplituden und Phasen enthält, die das Signal bilden. Die räumlich-dimensionalen Pulse verbessern die Klassifizierungsleistung der nachfolgenden SNN-Population.

Die Validierung der vorgeschlagenen Kodierung wurde anhand drei ausgewählter Signalverarbeitungsanwendungen durchgeführt, z. B. die Erkennung von Radarinterferenz, die Phasenerkennung und die Schätzung des Einfallswinkels zur Lokalisierung von Objekten. Die Ergebnisse zeigen, dass die R&F Kodierung existierende Kodierungsmethoden in Bezug auf die Effizienz und die Genauigkeit der nachfolgenden SNN-Population übertrifft. Die vorgeschlagenen SNN-Architekturen zeigen eine vergleichbare Leistung zu aktuellen Lösungen und bieten gleichzeitig die Fähigkeit zur Stream-Verarbeitung, einer schnellere Klassifizierung, die Fähigkeit zur Rauschunterdrückung und die Verwendung von spärlichen Pulsen.

Die entwickelten Architekturen belegen, dass SNNs mit R&F Kodierung wertvolle Alternativen zu herkömmlichen Methoden sind und zum Verständnis der Informationsverarbeitung in Neuronen Populationen beitragen. Die Fähigkeit, Sensorsignale direkt zu verarbeiten, ermöglicht eine hocheffiziente hardwarebeschleunigte Neuromorphe Signalverarbeitung in verschiedenen energiebeschränkten Systemen.

Acknowledgments

I am very thankful to Prof. Alois Knoll for his supervision throughout my research and his support for the close cooperation between academia and industry. He provided valuable advice and guidance so that I could successfully complete the research project. His ideas, discussions, and critical feedback have been a great source of motivation.

Also, I would like to thank you, my industrial supervisor and mentor, Dr. Cyprian Grassmann, for all the open discussions and the opportunity to research the field of spiking neural networks in the KI-ASIC Project. I am very grateful to Infineon Technologies AG and the German Ministry of Education and Research (BMBF) for supporting the KI-ASIC project(Grant Number 16ES0992K) and this research.

Special thanks go to my close colleagues Dr. Daniel Auge, Hendrik Lehmann, and Oliver Emonds for the intensive collaboration across many projects with valuable discussions. Also, I want to thank my university colleagues Nico Reeb, Robin Dietrich, Negin Karimi, and Javier López Randulfe for all the thoughtful teamwork and support. I am thankful for the close collaboration with Felix Kreutz, Pascal Gerhards, Daniel Scholz, and Jiaxin Huang from Infineon Dresden. We formed a team with my colleagues that could create thoughtful discussions and collaborations. However, I thank all my colleagues at Infineon, TUM, TUD, and OTH for their support, feedback, and discussions during my research.

Additionally, I thank Seifeddine Saadani and Daniel Scharf from the OTH Amberg-Weiden for their supportive data collection. Overall, I would like to thank my colleagues at the University and Infineon for the enjoyable environment during the challenging times of a global pandemic, lockdown, and “home office”. Online discussions and conferences were part of the daily work.

Last, I thank my family for their endless private and professional support. Troubled times come and go, but the family remains. Special thanks go to my wife, Katharina, for her unlimited support and assistance in troublesome times. Thank you for everything.

Contents

List of Figures

List of Tables

List of Acronyms

RQ Research Question

LIDAR Light Detection and Ranging

radar Radio Detection And Ranging

MIMO Multiple-Input and Multiple-Output

MMIC Monolithic Microwave Integrated Circuit

IF Intermediate Frequency

SNR Signal-to-Noise Ratio

FFT Fast Fourier Transform

DFT Discrete Fourier Transform

ADC Analog-to-Digital Converter

FMCW Frequency Modulated Continuous Wave

OFDM Orthogonal Frequency-Division Multiplexing

RCS Radar Cross-Section

RDM Range-Doppler-Map

STFT Short-Time Fourier Transform

CFAR Constant False Alarm Rate

MUSIC Multiple Signal Classification

ESPRIT Estimation of Signal Parameters via Rotational Invariance Techniques

AWGN Additive white Gaussian noise

SD standard deviation

AI Artificial Intelligence

NN Neural Network

ANN Artificial Neural Network

SNN Spiking Neural Network

SNN pulsende Neuronale Netze

R&F Resonate-and-Fire

ReLU Rectified Linear Unit

ML Machine Learning

MLP Multilayer Perceptron

MSE Mean-Squared-Error

RMSE Root-Mean-Squared-Error

EPSP Excitatory Postsynaptic Potential

IPSP Inhibitory Postsynaptic Potential

CNN Convolutional Neural Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

NLL Negative Log Likelihood

TTFS Time-To-First-Spike

HSA Hough spiker algorithm

BSA Ben's spiker algorithm

TC Temporal Contrast

CJ Current Injection

ROC Rank-Order Coding

TBR Threshold-Based Representation

SF Step-Forward

ISI Inter-Spike-Interval

STDP Spike-Timing-Dependent Plasticity

BPTT Back-Propagation Through Time

TBPTT Truncated Back-Propagation Through Time

LIF Leaky-Integrate-and-Fire

LI Leaky-Integrator

HH Hodgkin-Huxley

JIT Just-In-Time

Notation

a, A	A scalar (real or integer)
$\underline{a}$	A scalar (complex)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A complex matrix
$\underline{\boldsymbol{A}}$	A matrix
$a(t)$	A temporally changing variable (continuous time)
$a[t]$	A temporally changing variable (discrete time)
a_i	The i-th element of vector $\boldsymbol{a}$, with the index starting at 1
$a_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$a^{(i)}$	Element within the i-th layer of a network
$a^{(i,j)}$	Element with contributions from layer i to layer j
a_{descr}	A variable with distinct description
$f(a)$	A function of a
$Re(\underline{a})$	Real part of complex variable $\underline{a}$
$Im(\underline{a})$	Imaginary part of complex variable $\underline{a}$

t	Time
Δt	Time interval
t^f	Firing times of a neuron
$v(t)$, $\underline{v}(t)$, $v[t]$	Membrane voltage of a neuron
v_{th}, ϑ	Threshold voltage of a neuron
v_{reset}	Reset voltage of a neuron
C	Membrane capacitance of a neuron
R	Resistance of a neuron
$\delta(t)$, $z[t]$	Spike event
$i(t)$, $i[t]$	Input current of a neuron
τ	Time constant of a neuron
b	Damping constant of a neuron
$\boldsymbol{W}$	Weight matrix containing the synaptic connection weights
f	Frequency
ω	Angular frequency
θ	Heaviside step function
η	Learning rate
$\mathcal{L}$	Loss function
ψ_{lin}, ψ_{sig}	Pseudo gradient
$\nabla_x f(\boldsymbol{x})$	Gradient of $f(\boldsymbol{x})$
e_n	Envelope of the response of neuron n
$\sigma(x)$	Sigmoid function
$S(x)$	Logistic sigmoid function
N_{xxx}	Number of xxx
P_{xxx}	Power of xxx

1 Introduction

Artificial Neural Networks (ANNs) are a cutting-edge technology in many modern applications. Due to their ability to learn non-linear correlations, they are superior to classical deterministic methods. Their use is established in applications including image and speech recognition, robotic control, and sensory processing. The first vehicle with a Neural Network (NN) automated road following system was published in 1988 [1] and fueled the interest in autonomous driving cars. Significant milestones of NNs are competing games like chess, Go, and arcade video games [2], [3]. Recent advances use natural language processing to interact with human beings, answer questions, generate creative content, and create executable software code. NN methods have become essential to modern computing systems and our daily life.

Deeper network architectures consisting of multiple layers are computationally more powerful, making data pre-processing less relevant. Since deeper networks consume immense energy during training and utilization, they do not apply to memory and power-limited edge devices [4]. The GPT-3 model is estimated to consume 358 kW [5], while the human brain has a peak energy consumption of 20 Watts [6]. Since power consumption includes the processing of every sensory input simultaneously, one has to consider that the brain consumes less energy for a specific task. The current discussion about global warming has increased the trend of reducing the carbon footprint of such trainable computer systems with approaches like Green AI [7].

ANN algorithms are also considered a fundamental element of autonomous driving, where the driving application should be real-time without cloud interaction. In particular, energy consumption is an increasing concern without compromising the driving range when processing and fusing sensor data from the camera, Lidar, geographic positioning, gyroscope, and Radio Detection And Ranging (radar) sensors. Current radar sensors consume around 3.3 Watts [8]. At the same time, in nature, bats use echolocation as a primary environmental sensor with a power cost of 0.67 Watts during rest, including signal transmission, receiving, and processing [9]. Since the bats synchronize their wing beat to the pulse transmission, they can further reduce power consumption [10]. Bat's sensing functionality is comparable to the general idea of radar, differing in the signal transmission by the bat's tongue instead of antenna elements. The sensors extract the object's location with the reflections of the transmitted signal, which are the concepts for radar and bat's echolocation. While bats use frequencies from 25 to 150 kHz, the automotive radar frequency is about 77 GHz. Two receiver channels are sufficient to sense the three dimensional environment by analyzing the differences in amplitude and phase [11]. Thus, while hunting, the bats use an increased pulse repetition frequency of up to 200 pulses per second [11], and at rest, they can passively listen to the sound of

targets [12]. Bats also experience the problem of signal interference and use frequency hopping, inspiring radar interference mitigation [13].

One promising biological-inspired network approach is the Spiking Neural Networks (SNNs) because of their advantages over prior NN. They are seen as the third generation of NN [14]. All generations use the same concept, where an activation function is applied to the multiplication of the input with real-valued weights and an additional bias. The weights and biases generate a set of hyperplanes in the n-dimensional space to separate regions into distinct areas.

The history of SNNs can be traced back to 1943, when Warren McCulloch and Walter Pitts proposed the first neuron model [15]. The neuron is called perceptron, a binary computational unit that uses the "all-or-nothing" behavior observed in the nervous system. The perceptron is the fundamental element of network architectures like the multi-layer perceptron, Hopfield nets, and Boltzmann machines [14]. Afterward, in 1952, Hodgkin and Huxley discovered the mechanism of the action potential that travels along an axon. They conducted experiments on the giant axon of the squid, which allowed them to accurately measure the electrical activity of a single neuron and build the foundation for more complex computational units. The second generation of NNs replaces the threshold activation function with non-linear functions, like sigmoid, tanh, or Rectified Linear Unit (ReLU). The universal approximation theorem states that second-generation networks can approximate any linear and non-linear function [16]. The support for learning techniques based on gradient descent, like backpropagation, is another distinguishing feature of this network generation [17]. The current state-of-the-art uses second-generation concepts while interpreting the biological firing rates in a step-wise representation. However, [18], [19] demonstrated that the visual processing speed conflicts with the information processing through firing rates since the individual neuron has a firing frequency below 200 Hz [20]. In this work, we name the second-generation networks ANNs since they are less biologically inspired.

In contrast to the floating point communication of the second generation, the third generation uses binary events to transport information. The main difference between the perceptron and the spiking neuron is the processing over time, meaning that the neuron dynamics have a local memory to remember short-term information. In general, there are many neuron models. However, from a connectionist or engineering perspective, models like the Hodgkin-Huxley are too computationally complex, and simpler versions such as the Leaky-Integrate-and-Fire (LIF) are more promising. In general, SNNs have the following advantages over ANNs:

- Temporal dynamics: These spiking neurons operate in the temporal dimension since the neuron dynamics have a local state that changes for incoming events and leaks historical information over time. [21], [22]

- Event-driven: SNNs process information asynchronously and only at incoming events, reducing the computational burden and making them more effective. [21], [23], [24]

- Efficient structure: Traditional ANNs use a layer-wise representation, while SNNs operate as a population where any connection could exist, leading to a lower neuron count. Thus, the output neuron can directly connect with the input neuron, improving the information flow but increasing the training complexity. [25], [26]

- Distributed processing: The sparse network structure and the binary communication allow distributed processing in multicore platforms like SpiNNaker [27] because of the reduced communication between the processing elements or between subnetworks which can be implemented in analog or digital hardware. [28], [29]

- Speed: The neurons can provide an early estimation of the results, which further improves over time. Also, the possibility to directly process the continuous input stream and communicate between neurons by binary events removes additional latency. In general, the sparse connectivity improves the speed of the SNNs. [24], [30]

- Robust: The threshold function of a spiking neuron operates as a noise filter, and a population of neurons has an inherent noise activity, which makes the network more resilient to noise. [31], [32]

- Hardware-efficient: The computational effort of the synaptic weights in digital hardware reduces from a multiplication to an addition due to the binary events between neurons. An electrical circuit often describes the biological neurons, enabling an efficient analog implementation. [33], [34]

- Adaptive & local plasticity: SNNs take advantage of local adaptivity mechanisms such as the spike rate adaptation to adapt to temporal changes and reduce the spike count. Adapting the synaptic weights by limiting the number of synaptic transmitters enables a local plasticity mechanism. In contrast to traditional backpropagation optimization, local plasticity provides a short-term and long-term adaptation that can even provide real-time local learning in neuromorphic hardware. [35], [36]

The information must be converted to spikes to gain from the advantages of SNNs because these networks use discrete events to represent and process information, unlike ANNs, which use continuous values. We call the transformation into spikes "encoding" and the conversion from spikes "decoding". Decoding is mainly essential in neuroscience to understand the activity of the brain. One impressive example is decoding the sensorimotor cortex's brain activity of persons with articulate speech loss [37]. A traditional ANN interprets the recorded spike activity and converts it into a written language such that people can "speak" again. This setup demonstrates that the current research has not fully understood the information content in spikes.

Thus, encoding information is essential for the overall system's efficiency during the architectural design of an SNN. Encoding accomplishes two tasks: representing the crucial information into spikes while removing unnecessary information. An incorrect encoding will lead to inaccurate and inefficient processing because too many spikes cost more computations, and there is no information to process without a spike event. Currently,

there is not yet a single solution that fits most applications. However, with this work, we contribute to this by proposing a frequency-selective encoding scheme for radar signals.

Inspired by the biological observation of bats and the promising lower energy consumption of SNNs, we investigate and develop spiking architectures for radar signal processing. Therefore, thinking about temporal spike events, developing new architectures, signal representations, and modifying established learning methods is required. The recent adaptation of the non-differentiable activation function of the SNNs enables the utilization of gradient-based backpropagation optimization [38], the gold standard for training ANNs [17]. Therefore, the main objective of this dissertation is to investigate the problem of signal-to-spike encoding for SNNs in radar signal processing. Specifically, we will build on the recent advances and apply frequency-selective encoding to applications such as interference detection, phase detection, and angle-of-arrival estimation. Our goal is to contribute to understanding SNN-based signal processing and demonstrating the conceptual advantages of SNNs compared to ANNs.

1.1 Research Questions and Scope

Main Research Question
How to process sensor signals with SNNs efficiently?

Biological observations inspire this overall research question because nature uses sparse, event-driven, and asynchronous information to increase the system's efficiency. Therefore, the motivation is to connect the SNNs directly to analog sensors by representing the compressed data in temporal spikes. The contribution of this thesis focuses on the conceptual design and analysis of SNNs that directly process the radar time-domain signal with a suitable signal encoding. We narrow the concepts to applications like radar interference detection, phase detection, and radar angle-or-arrival estimation. Accordingly, we break down the overall Research Question (RQ) into the following four questions:

One key element of spiking networks is transporting information through sparse spike events, usually represented as binary events. Nowadays, most data exists in a floating or fixed point representation, and traditional networks are designed to use this kind of data. While the human brain uses spikes, we adjust our data representation to a human-readable form. Due to evolution, nature has developed efficient methods to convert information, and there is currently no single method to convert any data into spatio-temporal spikes. Consequently, we define our first RQ:

Research Question 1
How to transform temporal data into spike representation for processing with SNNs?

Current state-of-the-art automotive radar extracts information about an object's range, velocity, and angle from the frequency and phase. The range corresponds to the frequency and the angle to the phase of multiple receive channels. Bats perform echolocation by extracting the phase differences of the received echo, which is an inspiration to directly extract the frequency and phase from a multi-tone signal and convert it into a spike

representation without additional pre-processing. Accordingly, we raised the following RQ:

Research Question 2
How can the frequency and phase of the received radar signals be represented in spike-events?

In order to increase the energy advantage of SNNs, the entire system must use sparse asynchronous processing. Therefore, it is crucial to design concepts and architectures which use fewer pre-processing steps and apply SNNs to temporal data. Additionally, energy and memory constrains limit near-sensor processing. To demonstrate the processing of the time-domain signal with frequency selective encoding, we select the application of interference detection while enabling stream processing without storing historical information. This application demonstrates the pattern separability of different NN architectures on a prediction and classification approach, answering the following RQ:

Research Question 3
Which NN and SNN architectures can detect outliers in the time-domain radar signals?

In the previous RQ, we focused on an application that detects patterns in the time-domain signal. Therefore, we extend it with an example application that uses both frequency and phase. In radar, the frequency mainly defines an object's distance, and the phase's information depends on the relative reference point. On the one hand, the phase change across consecutive chirps of one transmitter and receiver pair carries the velocity of an object. On the other hand, the phase change between receiving antennas provides an estimate of the angle-of-arrival. The velocity estimation needs an additional temporal memory to compare the differences between two consecutive chirps, and the information can also be received by analyzing the changes between the chirps. More challenging is the estimation of the angles. Inspired by biological methods like the echolocation of dolphins or bats, we investigate solutions for a spiking angle-of-arrival estimation in radar applications. Thus, we define the following RQ:

Research Question 4
How to perform the angle-of-arrival estimation with SNNs on time-domain radar signals?

1.2 Contributions

Some parts of this thesis have already been presented at international peer-reviewed journals or conferences. We provide the following journal publication with an overview of existing methods to convert signals or images to spike events. The publication presents a taxonomy of the different encoding methods, their biological inspiration, and example applications in engineering. It is the basis to answer the first RQ 1, and its content is reflected in Section 3 with an extension of the Resonate-and-Fire (R&F) encoding.

1. Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. **A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks.** *Neural Processing Letters (2021).* [39]

The first radar-based application is the detection of outliers on the time-domain signal. The first concept with a traditional ANN was published at a conference, a basis for developing SNN outlier detectors. The semi-supervised trained system consists of a supervised, trained predictor and a threshold-based classifier. Its content is included in Chapter 4, answering RQ 3 partly.

2. Julian Hille, Daniel Auge, Cyprian Grassmann, and Alois Knoll. **FMCW radar2-radar Interference Detection with a Recurrent Neural Network.** In *2022 IEEE Radar Conference (RadarConf).* [40]

Furthermore, RQ 3 also asks for SNN architectures, and therefore, we investigate different encoding methods and architectures to solve the interference detection with an SNN. In this thesis, Chapter 2.1.3 demonstrates an extended analysis of the architecture search and validation with synaptic and spike sparsity methods and an investigation of the different R&F rest and spike functions. The following publication at an international conference demonstrates the generated concept and the best-performing SNN architecture.

3. Julian Hille, Daniel Auge, Cyprian Grassmann, and Alois Knoll. **Resonate-and-Fire Neurons for Radar Interference Detection.** In *2022 International Conference of Neuromorphic Systems (ICONS2022).* [41]

Apart from interference detection, we use a phase detector system to investigate the differences between the two current injection encoding methods. We describe the theoretical background of the encodings in Chapter 3 and answer RQ 2. The spike encoding with LIF or R&F neurons uses the neuron's different inherent properties and reacts differently to noise or signal superpositions. The comparison of the encodings is shown in Chapter 5, and we have published the analog hardware implementation of the phase detector in an international journal.

4. Hendrik M. Lehmann, Julian Hille, Cyprian Grassmann, Alois Knoll, Vadim Issakov. **Analog Spiking Neural Network Based Phase Detector.** In *IEEE Transactions on Circuits and Systems I: Regular Papers.* [42]

The following contributions are not directly a part of this thesis but have contributed significantly to these publications in the research field of SNN concepts, applications, and hardware realizations. First, we demonstrate speech recognition with R&F encoding and compare it with the state-of-the-art. The end-to-end architecture uses directly the digital audio signal to detect keywords like "off", "on", and more. We compare the complete SNN architecture with the current state-of-the-art Mel-spectrum analysis. This work was presented at an international conference.

5. Daniel Auge, Julian Hille, Felix Kreutz, Etienne Mueller, and Alois Knoll. **End-to-End Spiking Neural Network for Speech Recognition Using Resonating Input Neurons.** In *2021 30th International Conference on Artificial Neural Networks (ICANN).* [43]

Further, we analyzed different binarization methods of sequential Range-Doppler-Maps (RDMs) to classify hand gestures with an SNN. The different spiking architectures work with varying forms of conversion of frame-based data. The temporal changes across the RDMs are classified with the inherent temporal dynamics of the spiking neurons. The publication was presented at an international conference, where the poster was rewarded with the best poster award.

6. Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. **Hand Gesture Recognition in Range-Doppler Images Using Binary Activated Spiking Neural Networks.** In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG2021).* [44]

Since this research is part of the KI-ASIC project [45], funded by the German Ministry of Education and Research (BMBF), we evaluated the usage of SNNs for automotive radar processing with the focus on a spiking Fourier transformation and the realization of a spiking Constant False Alarm Rate (CFAR) algorithm. Besides, we provide some future perspectives on where SNNs can be used in the radar signal processing pipeline. The following joint journal contribution reflects knowledge gained by our research.

7. 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. **Automotive Radar Processing with Spiking Neural Networks: Concepts and Challenges**. Frontiers in Neuroscience: 414. [46].

Besides, we have investigated other automotive-related applications, like detecting the exact motor position in an electrical engine. The motor position sensors are expensive, and a precise prediction of the motor position provides better cost-efficiency. Thus, we investigated using a spiking network to predict the position, a regression problem similar to the radar signal prediction described in Chapter 4. We submitted this work to an international conference.

8. Felix Kreutz, Daniel Scholz, Julian Hille, Jiaxin Huang, Florian Hauer, Klaus Knobloch, Christian Georg Mayr. **Continuous Inference of Time Recurrent Neural Networks for Field Oriented Control**. In 2023 IEEE Conference on Artificial Intelligence. [47]

Additionally, the following contributions have been gained from the research of this thesis through the conceptual investigation of neuron dynamics and encoding methods. The findings of these conference publications demonstrate analog circuits of the LIF neuron for the automotive application with optimized dynamics. While these publications are not a direct part of this thesis, they contribute to understanding the hardware limitations and the concept requirements to enable SNNs on dedicated analog hardware. The first publication focuses on implementing an Izhikevich neuron for a logic gate application. We test the usability of the circuit under automotive circumstances like temperature range in real measurements. This work was published at an international conference but is not elaborated on in this thesis.

9. Hendrik M. Lehmann, Julian Hille, Cyprian Grassmann, Vadim Issakov. **Spiking Neural Networks based Rate-Coded Logic Gates for Automotive Applications in BiCMOS**. In *2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*. [48]

The next work investigates the analog implementation of a LIF neuron with an invariant spike generation by a new refractory period mechanism. Real measurements demonstrate the stability of the output spike events. It was published at an international conference.

10. Hendrik M. Lehmann, Julian Hille, Cyprian Grassmann, Vadim Issakov. **Leaky Integrate-and-Fire Neuron with a Refractory Period Mechanism for Invariant Spikes**. In *2022 17th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. [49]

Additionally, we investigated the R&F neuron in analog hardware to encode analog signals directly. This work provides the first investigation of the concepts explained in this thesis by enabling the direct conversion of the signals to spikes instead of using complex and non-linear Analog-to-Digital Converters (ADCs). This work is published in an international open-access journal.

11. Hendrik M. Lehmann, Julian Hille, Cyprian Grassmann, Vadim Issakov. **Direct Signal Encoding with Analog Resonate-and-Fire Neurons**. In *IEEE Access*. [50]

1.3 Structure

As shown in Figure 1.1, the thesis is divided into eight chapters. The first chapter motivates and describes the RQs to resolve the identified knowledge gaps. Subsequently, a brief description of the publications that are either directly or indirectly related to this thesis is provided. Chapters 2.1 and 2.2 set the basic knowledge about radar and SNNs. We provide the theoretical principle of radar and state-of-the-art signal processing, focusing on interference detection and angle-of-arrival estimation. Afterward, Chapter 2.2 covers the basic biological-inspired neuron models, the synapse model, and training principles with pseudo-gradient backpropagation and evolutionary optimization.

Chapter 3 sets the fundamental methodology of signal-to-spike encoding. It reviews signal encoding techniques and describes the proposed R&F current injection method that is the base of all successive chapters. During the Methodologies & Experiment chapters, we demonstrate three radar-related applications that use R&F encoding. In Chapter 4, we design a radar-to-radar interference detector. Therefore, we compare a prediction-based and a pattern classifier approach with traditional networks and SNNs. We further optimize and analyze the R&F encoded SNNs by different spike functions, various reset methods, time constants' optimization, spike count reductions like spike regularization, and network sparsity. Before discussing the experiments' results, we analyze the noise robustness of the most promising architectures.

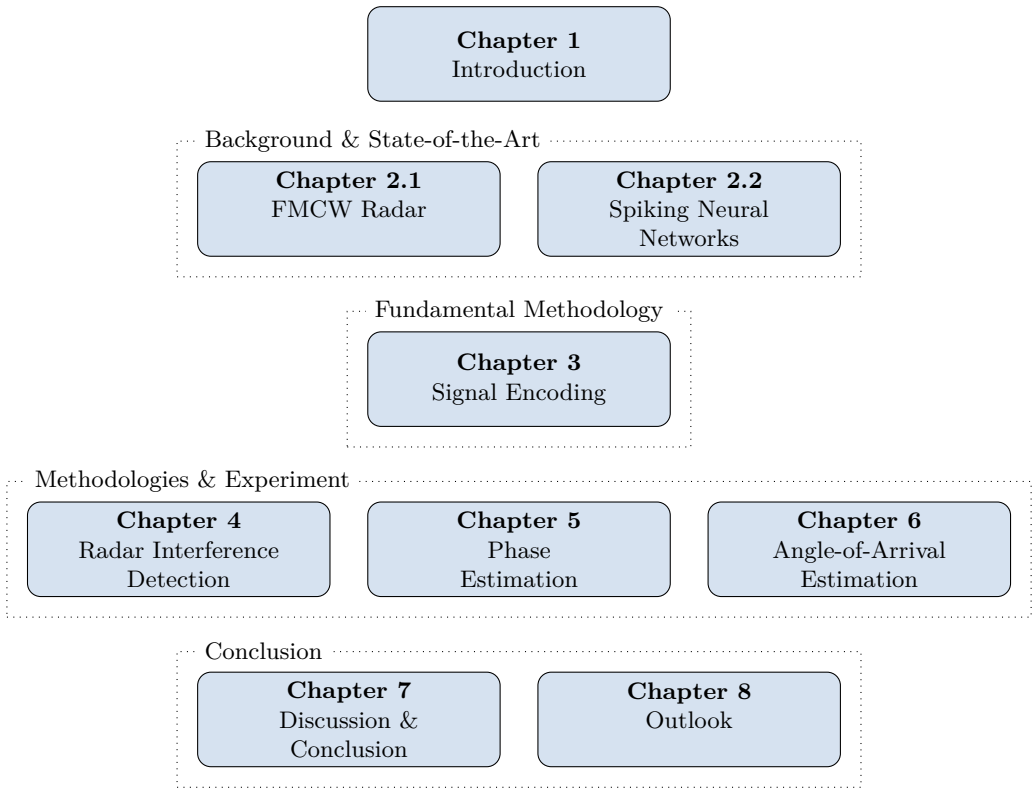

Figure 1.1: Structure of this thesis

In Chapter 5, we use the phase-dependent variant of the R&F neurons to implement a phase detector. We differentiate between the LIF and R&F encoding and describe their system's influence. After the software-based proof-of-concept, we realize the LIF-based network in analog hardware.

In the last chapter of the methodologies, we describe how SNNs can detect the range and angle information on the ADC-samples of the radar sensor. The concept demonstrated in Chapter 6 uses the R&F neurons to encode the radar signals and detect the objects' angles. Various experiments prove the angle estimator's application compared to the state-of-the-art. Afterward, the chapter ends with individual discussions of the findings.

Chapter 7 provides a discussion of the overall findings, implications, industrial relevance, and limitations of the methodologies. Finally, Chapter 8 suggests an outlook on the future research directions and topics of signal processing with SNNs.

2 Background

The background and state-of-the-art of this thesis can be separated into two distinct chapters of radar and SNN fundamentals. In the radar background, we explain the general concept of environmental sensing and the specific properties of the signal during radar interference. The SNN chapter focuses on the biologically inspired neuron models and the synapses we use in the applications. Besides, we describe and motivate the surrogate gradient learning to optimize the network weights.

2.1 Automotive Radio Detection and Ranging Sensor

Radar sensors are one of the key elements for autonomous driving and driver assistance systems. These sensors sense the environment by reflecting high-frequency signals and extracting the range, velocity, and angle-of-arrival for objects independent of lighting and weather conditions. This technique is based on the findings about the reflectivity of electromagnetic waves on metal surfaces [51]. The developer Christian Huelsmeyer is seen as the explorer of the radar system with his patent in 1904 [52], where he described a method to detect moving metal-based objects using electromagnetic waves. Today, radar sensors have improved performance, size, price, and production volume. The sensors are optimized for specific applications like the military, automotive industry, and human sensing [8], [44], [53], [54]. The first ideas for automotive radar state back in the 1960s for collision avoidance applications [55]. Research mainly focuses on the increasing carrier frequency from the MHz range to the 76 GHz systems. In 1998 the first commercial 76 GHz automotive radar was available in the Mercedes-Benz S class built by Macom in the USA [56].

The newest trend is 3D imaging radar [57] and Multiple-Input and Multiple-Output (MIMO) sensors for high-resolution detection with up to 1728 virtual channels by using 36 transmit and 144 receive antennas through cascading of 12 Monolithic Microwave Integrated Circuits (MMICs) [58]. The MIMO principle uses multiple transmit and receive antennas to form a bigger virtual aperture to improve angular resolution in azimuth or elevation. The primary motivation behind image-like radar systems is the replacement of passive sensing systems, enabling robust and weather resistance sensing with high angular resolution. In addition to the position information, the radar can provide the velocity of an object by detecting the object's changes during a short time. Currently, the automotive industry uses Frequency Modulated Continuous Wave (FMCW) mainly due to the lower complexity, high reliability, high performance, high production volume, and low costs. Meanwhile, research evolved to alternative digital modulation schemes such as Orthogonal Frequency-Division Multiplexing (OFDM) [56].

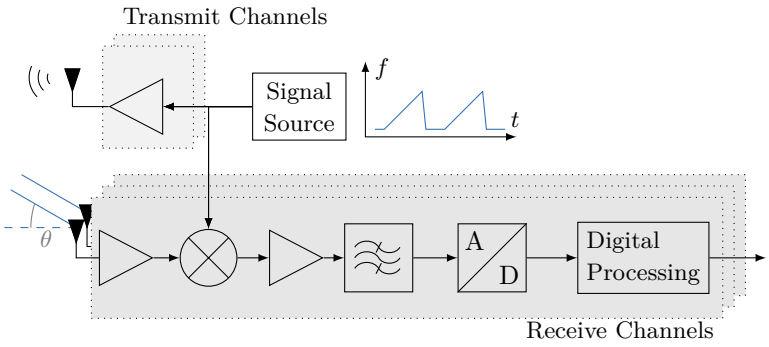

Figure 2.1: MMIC block diagram: Simplified representation of the FMCW radar function of a single transmit and receive channel. Separation of the transmit and receive pipeline. Modified from the publication [40].

In automotive, the number of sensors increases with the higher level of automation. Currently, radar applications are collision avoidance, adaptive cruise control, blind spot detection, cross-traffic alert, and lane departure warning [59]. Therefore, a front position radar cannot detect cross-traffic in a parking lot situation of the car's backside. Since the antenna layout of the radar sensor defines its application, we differentiate between short, mid, and long-range radar. The long-range focus is on $10 - 250$ m, mid-range on $1 - 100$ m, and short-range on $0.15 - 15$ m, classified by [60]. Thus, the number of sensors per vehicle increase drastically to support the driver with specialized sensors. Additionally, there are approaches using the existing radar for car-to-car communication [61].

We describe the basic concept behind the current state-of-the-art automotive FMCW radar, the signal processing pipeline, and the interference effect between sensors. We set the focus of the chapter to the understanding of the applications. Also, we used a company's internal simulator for synthetic radar data generation based on the general concept of the mathematical description.

2.1.1 Concept

Radar is the standard in the automotive industry to detect the surrounding environment, and it uses high-frequency reflections to extract the range, velocity, and angle of the targets.

Many modulation schemes modify the amplitude, phase, or frequency [62]. Due to its small size, high resolution, low cost, mass production, and reliability, FMCW has prevailed in the automotive sector [63]. An FMCW sequence uses consecutive linear frequency chirps with adjustable slope, bandwidth, and timing, affecting the range and velocity resolution. Figure 2.1 depicts a block representation of the MMIC for FMCW radar where the signal source generates a sequence of frequency chirps, like demonstrated in the figure. A ramp is defined by its timing of idle, ramp, and flyback, where the ramp time defines mainly the properties of the MMIC limit, the range resolution, and the flyback and idle time. The idle time ensures the signal settles before a new chirp after the

jump back to zero frequency. Often the signal generated mixes the carrier signal and the modulation signal. Currently, 77 GHz is the default carrier frequency in the automotive industry. Nevertheless, there are also 24 or 60 GHz [64], [65]. The signal is forwarded to the different transmit channels, and an optional phase shifter enables beamforming which is not shown here. Before sending the signal through the antenna, it is amplified because the signal power influences the maximum range of detectable objects. The receive signals are amplified and mixed with the transmit signal to retain the targets' Intermediate Frequency (IF). An amplifier and bandpass filter preprocess the signal before the ADC to reduce alias effects. Afterward, the digital filter improves the signal, and the valid component of the signal are separated, meaning that the time between the chirps does not carry information about the targets and can be removed. The targets are extracted via signal processing from the time-domain signal, as described in Section 2.1.2.

To simulate the radar functionality and understand interference, we model the function using the mathematical description inspired by [66], [67]. Accordingly, we define the time-dependent frequency sweep of the transmitter in a phase-modulated signal form:

$$s_{\mathrm{TX}}[t] = A_{\mathrm{TX}} \cos\left(\phi[t]\right) \tag{2.1}$$

$$\phi[t] = 2\pi \int_0^t \underbrace{\left(f_c + \frac{B}{T_m} t\right)}_{\text{instantaneous frequency}} dt = 2\pi f_c t + \pi \frac{B}{T_m} t^2 + \varphi_0, \tag{2.2}$$

where the frequency increases linearly during the chirp duration T_m from the carrier frequency f_c by the chirp bandwidth B. The transmitter signal can have a phase offset defined by φ_0 for beam-steering. The targets reflect the transmit signal and return a delayed wave based on the round-trip-delay-time $t_d = \frac{2r}{c}$, proportional to the distance r between the sensor and the target. The received signal for a single target

$$s_{\mathrm{RX}}[t] = \alpha \cos\left(2\pi f_c \left(t - t_d\right) + \pi \frac{B}{T_m} \left(t - t_d\right)^2\right) \tag{2.3}$$

alters by the transmission path attenuation α. Hence, the real amplitude of the received signal depends on the antenna gain, transmit power, target distance, and target Radar Cross-Section (RCS). The RCS depends on the size, reflectivity of the material surface, angular direction, and geometric shape [68]. [69] provides a comprehensive overview of RCS and highlights the difficulties in defining it. The well-known radar equation approximates the receive power P_R of a known target at distance R with an RCS σ [60]:

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi)^3 R^4 L} \sigma, \tag{2.4}$$

where L combines losses in the sensor, P is the power of the transmitter T or receiver R, and G is the gain. Hence, an increase in the distance reduces the received power and makes it challenging to detect less reflective small objects at a far distance, like a child at 200 meters.

As shown in the block diagram in Figure 2.1, the mixer multiplies the received echo signals with the transmitter signal. The mixing results in the sum and the difference between the receive and transmit signal. Since the sum component is in the order of twice the carrier frequency, it is suppressed by the electronic components and the band-pass filter. We assume that a real baseband chain is implemented without the imaginary component. Thus, the IF at the mixer output can be written as:

$$s_{\mathrm{IF}}[t] = \frac{1}{2} A_{\mathrm{IF}} \cos\left(2\pi \underbrace{\frac{B t_d}{T_m}}_{=f_b} t + 2\pi f_c t_d - \pi \frac{B}{T_m} t_d^2 \right), \tag{2.5}$$

where f_b is the beat frequency of the corresponding target. Without loss of generality, the term $\pi \frac{B}{T_m} t_d^2$ can be removed because this phase rotation is insignificant compared to $2\pi f_c t_d$. This simplification leads to the following:

$$s_{\mathrm{IF}}[t] = \frac{1}{2} A_{\mathrm{IF}} \cos\left(2\pi f_b t + 2\pi f_c t_d\right), \tag{2.6}$$

which shows that minor changes in the distance across consecutive chirps mainly affect the phase because of its dependency on the carrier frequency f_c. The velocity information of a target can be determined by the Doppler effect, which describes the frequency shift of a wave when the receiver moves relative to the transmitter. In Equation 2.3, we neglect the Doppler shift because the short ramp duration of FMCW is not affected by the Doppler effect, and the position during one chirp is constant [70]. Thus, the distance change across multiple chirps contains the velocity information encoded in the phase of the object's frequency, as shown in Equation 2.6.

The range resolution is the ability to distinguish two close objects. The maximum resolution $d_{res} = c/2B$ depends only on the chirp bandwidth. The maximum velocity depends mainly on the ramp time of a single chirp $v_{max} = \lambda/4T_c$, while the resolution is affected by the number of chirps defining the duration of a radar frame T_f, leading to $v_{res} = \lambda/2T_f$.

In contrast, the maximum angle θ_{max} depends only on the physical properties of the antennas and the spacing l between the antenna elements:

$$\theta_{max} = \sin^{-1}\left(\frac{\lambda}{2l}\right). \tag{2.7}$$

Figure 2.1 indicates the angle of the incoming waves to the receiver antennas, where the distance between the object and the antennas introduces a slight delay. The individual distances between the targets and the antennas is defined by $lsin(\theta)$, introducing a phase difference.

The angular resolution defines the separability between two objects at the same distance. In the literature, the antenna aperture is a common limitation described with the Rayleigh criterion, a heuristic condition for the minimum distance between two separable light

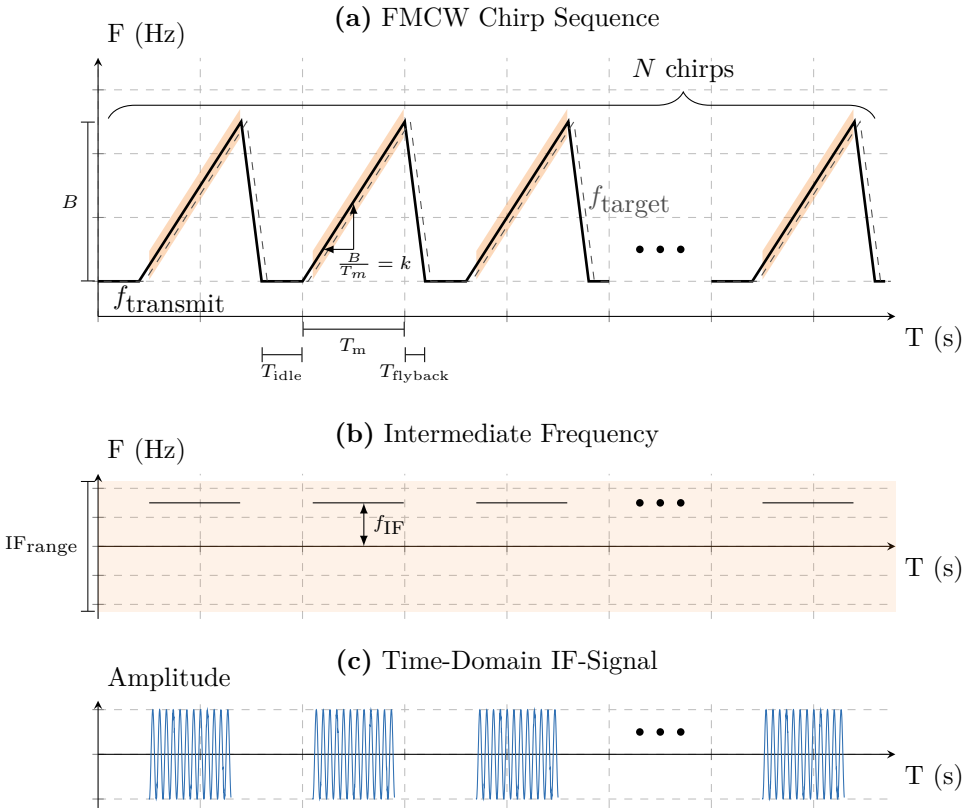

Figure 2.2: FMCW concept: Radar signals at different stages of the processing chain. (a) represents the frequency modulated chirp sequences in the instantaneous frequency of a FMCW radar. The transmit signal (solid) is reflected by an exemplary target (dashed), which stays within the IF bandwidth. (b) indicates the target frequency over time, proportional to the target's distance. (c) is the corresponding time domain signal after the mixer and low-pass filter.

sources [71]. The Rayleigh criterion defines a correlation between the size of the aperture and the angular resolution, which means a bigger antenna aperture leads to a higher resolution. For an image like radar, increasing the number of antennas is necessary. Considering the lateral separability between two lanes of 3.75m at a distance of 200m requires an angular resolution of approximately 1.1° which would need 100 antenna elements with a single array [72]. Also, the angular resolution of 1° is a requirement for level 4 and 5 autonomous driving. One solution to reduce the number of antennas is the idea of virtual antennas, which is the multiplication of transmit and receive antennas to generate a virtual representation of the antenna aperture and increase the angular resolution [73]. Such systems are also crucial in wireless communication and are called MIMO [74]. The phase difference between the receivers is proportional to the direction of arrival. In Figure 2.1, we demonstrate the simplified received waveform from a target. The difference in the traveling time of the wave to both antennas defines the angle.

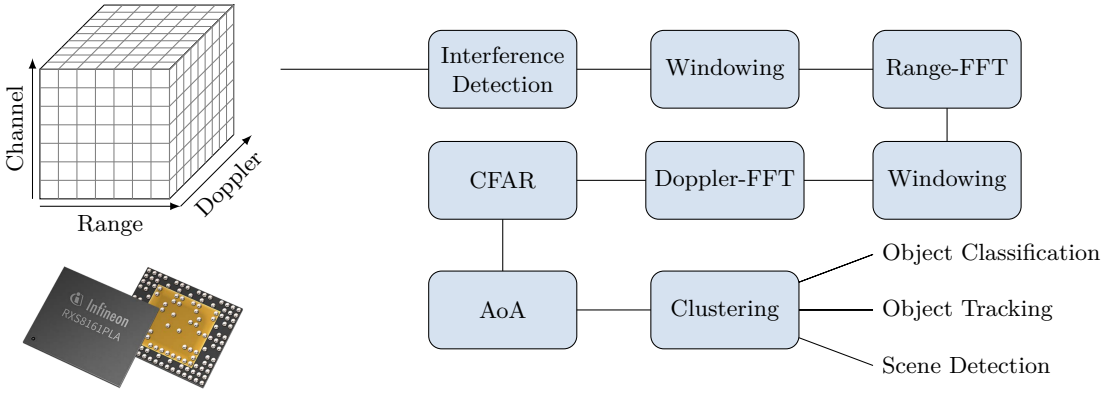

Figure 2.3: Radar signal processing chain: State-of-the-art signal processing chain of a radar sensor. The radar cube represents the dimensionality of the input data. The first windowing applies to the Range dimension, while the second to the Doppler dimension. AoA stands for angle-of-arrival estimation. The image of the MMIC is taken from [75].

Figure 2.2 demonstrates an example of the radar signals at the different stages. The transmit signal is shown in (a), indicating the typical parameters to define a ramp. The bandwidth B sets the differences between the minimum and maximum frequency within the ramp time T_m, defining the slope of the ramp. The orange marked area indicates the valid section used for signal processing. The example target leads to a round-trip delay that introduces a temporal shift of the transmit signal indicated by the dashed line. Mixing the difference between the transmit and receive signals defines the constant target frequency, IF, represented in (b). The signal is a superposition of the constant target frequencies proportional to the target range. The IF range defines the maximum and minimum frequencies supported by the receiver chain filters. A negative frequency can only appear if another sensor has the same chirp sequences or the echo of an object is too far away; both scenarios are implausible. In the last plot of Figure 2.2, we depicted the time-domain IF signal with a superposition of target frequencies. In this work, we apply mainly the algorithms on the time-domain signal, which corresponds to the shown signal.

2.1.2 Radar Signal Processing

We define radar signal processing as the step to extract the object's position in the three-dimensional space regarding the sensor and velocity information. Figure 2.1 demonstrates the individual block of an MMIC where the outputs are the ADC samples of the received signal. In the following, we will describe the subsequent steps to extract the information about the targets.

Current state-of-the-art radar signal processing assumes the ADC samples are stored in a three-dimensional matrix called a radar cube [72], [76]. The dimensions are range (fast-time), Doppler (slow-time), and channel, as shown in Figure 2.3. Also, the figure demonstrates a signal processing pipeline and presents an example chip of Infineon Technologies AG.

The first step is detecting interference in the time domain because interference affects the time-domain samples, leading to increased noise in the frequency domain, making it challenging to detect it [77]. The detected interference can either be mitigated or be used to trigger the stop of the processing since it is not safe. In Section 2.1.3 and Chapter 4, we will further describe the effect of interference and the state-of-the-art detection methods. Afterward, a windowing function across the range dimension removes artifacts by reducing the leakage problem. In non-periodic data, the signals are spread across the complete frequency range during the digital Discrete Fourier Transform (DFT), called the leakage problem [78]. Thus, the signal is preprocessed by a hanning window which reduces the ends of the data to zero. Then, we apply the first Fast Fourier Transform (FFT) across the fast-time or range-dimension of the radar-cube. This step extracts the individual frequencies from the IF signal since the echoes from targets are constant frequencies with varying amplitudes.

Two objects at the same distance can be resolved by analyzing the velocity. The velocity estimation extracts minor position changes across consecutive chirps decoded in the individual distances' phase. Before we apply the second FFT, we use a windowing across the Doppler dimension. The sinusoidal superposition of the phase changes of consecutive chirps contains information on the objects' velocity. An object that exceeds the maximum velocity can lead to ambiguous velocity information. While the velocity estimation performs well for longitudinal movements, it suffers from radial position changes [79]. We have generated the RDM, where the amplitude indicates a target by its reflectivity. The CFAR algorithm differentiates the background noise from targets through an adaptive threshold. The threshold depends on the noise level of the surrounding cells by cell-averaging, ordered-statistic, or other CFAR algorithms [80]. The previous signal processing steps are applied to each individual channel.

The last signal-processing step of the data extraction part is the angle-of-arrival estimation. To reduce the computational complexity, latency, and power consumption, we limit the angle estimation only to the detected targets from the CFAR algorithm. The angle estimation is the most challenging step since the number of antennas is limited. For imaging radars, achieving a high angular resolution in azimuth and the elevation direction is necessary. The elevation is essential to differentiate between a truck and a sign where the car can pass below.

One of the simplest angle estimation methods is the Bartlett beamformer, which maximizes the angular spectrum by exploiting the spatial diversity of the antenna array to extract the different signal delays. The method returns the position of the largest maxima as the estimation. In general, beamforming is a technique for spatial filtering. For a uniform linear array, the FFT can implement these spatial filter weights, which is computationally efficient due to the reuse of the FFT accelerator from the previous two FFT steps [81].

A more advanced angle estimation method is called Multiple Signal Classification (MUSIC) [82], which is part of the high-resolution category, obtaining the eigenvectors through constructing the covariance matrix and the eigenvalue decomposition. The method can accurately estimate the angles of multiple signals arriving from different directions. The largest eigenvalues indicate the signals, while smaller eigenvalues indicate

noise. Then the algorithm defines the spatial spectrum where peaks correspond to the angle-of-arrival of the targets. A computationally more efficient method is the Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT), which uses the rotational invariance among the signal subspace [83]. Therefore, differences in the subarrays create the matrix for eigenvalue decomposition. From the eigenvalues, signal parameters like the angle can be estimated.

However, all these deterministic methods suffer from computational complexity, the necessity of prior knowledge of the number of sources, or the low performance under noisy conditions. With the advances in machine learning, the idea of angle-of-arrival estimation with NN received interest. Three architectures have been shown to solve the angle estimation task. The first is a Multilayer Perceptron (MLP) consisting of multiple layers of sigmoid neurons [84]. The network receives a snapshot containing the channel dimension of a single detected range and velocity sample-point of the radar cube as input. Since the network predicts objects within $\pm 60^\circ$ with a resolution of 0.1°, it consists of 1201 output neurons with a softmax output function for categorical distributions. With extensive classifier training, the results demonstrate a super-resolution with limited antennas.

Another architecture inspired by computer vision research is called Convolutional Neural Network (CNN). The input is again a snapshot of the covariance matrix or after the eigenvalue decomposition [85]. [86] demonstrates a CNN with four convolutional layers and four fully connected layers with ReLU activations. The proposed network consists of 28.2 million trainable parameters, which exceeds the memory of every embedded processor. Another variant of a fully connected CNN architecture is demonstrated by [87]. [88] highlighted the reduced number of operations with a complex-valued CNN for angle estimation while receiving the preprocessed RDMs.

The autoencoder is an architecture that encodes the input into a latent representation, and the decoder extracts the original input with the possibility to perform noise suppression or error correction. [89] presented an autoencoder architecture based on six fully connected layers with a ReLU activation function. The proposed architecture is combined with MUSIC to improve the noise robustness and angle estimation. [72] focuses on a quantitative comparison of deep learning with traditional angle-of-arrival methods.

However, the computation complexity of this NN-inspired method is high because of the size of the networks. An often neglected part is the network training with a significant amount of simulated or measured data. In the subsequent steps, tasks like object classification, clustering, or tracking can be solved where also NN approaches are valuable [44], [90]–[93].

2.1.3 Radar Interference

The interference phenomenon appears when at least two waves move simultaneously through a medium. The combination of the waves can produce constructive or destructive interference, which means that the resulting amplitude is greater or lower. In the extreme case, destructive interference can lead to the complete removal of a signal which is the basic principle of noise-canceling headphones or microphones [94]. Such destructive

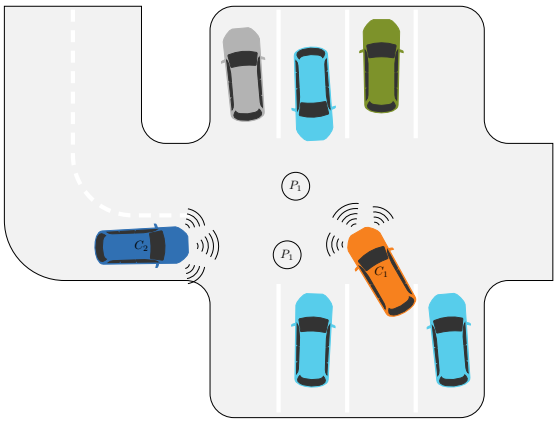

Figure 2.4: Automotive interference scenario: Typical parking lot situation with walking pedestrians and moving cars. Car C_1 interferes with car C_2 and blinds pedestrians because of their low reflectivity. The car shape is modified from [99].

interference only occurs when two signals have the same frequency and amplitude with a relative phase of 180°. The opposite is constructive interference, where the amplitude doubles when the phase matches between the signals. Thus, interference effects depend on the frequency, amplitude, and phase of the individual signals in the superposition.

In contrast to intentional noise reduction, interference is a significant problem for the performance of radar processing. Interference can occur due to internal or external effects. Internal effects are the crosstalk between channels or the leakage of the transmitter signal to the receiver channels. External effects are multipath-reflections or other sending transmitters in the same frequency range. The radar sensor is affected by interference with a degradation of the signal power, and the performance degrades. Thus, increasing radar sensors also increases the likelihood of interference which correlates with the interest in research projects like MOSARIM and IMIKO [95], [96]. Note that interference could also be seen as a possible attack to deactivate the radar sensor of autonomous vehicles. The first method is jamming [97], which saturates the receiver with noise by active interference that deactivates the sensor. The second method, spoofing, replicating, and retransmitting the radar signal with wrong information, leads to artificial targets [98]. Unwanted interference between sensors can occur in various situations, like highways, parking lots, intersections, and more. Figure 2.4 demonstrates a parking lot situation where detecting pedestrians to prevent accidents is crucial. Since the sensor generates the transmit signal at a random time, the likelihood of interference is low but not neglectable, especially with the increasing number of sensors. Thus, we focus in the following section on the radar-to-radar interference to understand the properties and be able to generate synthetic interference data that are used in Chapter 4.

As mentioned, interference is a superposition of the target reflections with an unwanted, unknown signal from another sender within the same frequency band. Here we use the notation of a victim and aggressor sensor, the affected and the stimulated device, respectively. We assume that both sensors are similar radar devices. Thus, we define the

transmit signal s_{TX} of the aggressor a, similar to Equation 2.1:

$$s^a_{\mathrm{TX}}[t] = A^a_{\mathrm{TX}} \cos\left(2\pi f^a_c t + \pi \frac{B^a}{T^a_m} t^2\right), \tag{2.8}$$

where the chirp slope $k^a = \frac{B^a}{T^a_m}$ defines the linear frequency change from the initial carrier frequency f^a_c. Since the signals of the aggressor sensor and the target's echos overlap in the free space, the IF of the aggressor signal in the victim sensor is:

$$s_{\mathrm{IF}}[t] = \frac{1}{2} \cos\left(2\pi f_i t + \pi \left(\frac{B}{T_m} - \frac{B^a}{T^a_m}\right) t^2 + \varphi\right). \tag{2.9}$$

The first term uses the difference of the carrier frequencies $f_i = f_c - f^a_c$ and the second the difference of the slopes. Therefore, we get the following instantaneous frequency:

$$f_{\mathrm{IF}} = f_i + \left(\frac{B}{T_m} - \frac{B^a}{T^a_m}\right) t. \tag{2.10}$$

There are two fundamentally different occurrences of interference called in-phase and out-phase. In-phase interference is when the aggressor frequency course is parallel to the victim's transmitter signal, meaning they have a different f_i but the same slope. With the same carrier frequency, the victim could interpret the aggressor signal as a target that does not exist, called a ghost target. The only possibility to detect in-phase interference is investigating the signal power, which we will explain later. In contrast to in-phase interference, out-phase is more likely and reduces the detectability of any target. Out-phase interference crosses the frequency courses, resulting in a triangular IF signal. Thus, the signal is spread across the complete frequency bandwidth and increases the noise floor because the power of interference is higher than for a signal reflection.

The likelihood of in-phase interference is neglectable small, as investigated by [100]. They defined the probability of in-phase interference by the maximum supported temporal shift, limited by the IF bandwidth. Thus, the probability of being affected by at least one interferer out of 1000 sensors with the same ramp scenario and starting frequency is 11.8%. Additionally, to the huge unrealistic amount of sensors, it is unlikely that the ramp scenarios of the sensors are the same. [101] showed that the probability of interference between two random radar sensors is 0.0665% which can be considered insignificant.

On the other hand, out-phase interference is a more common effect that drastically increases the noise. Again, [101] calculated that the probability of interference between two signals with a relatively uniformly distributed phase is 99.93%. However, the probability correlated directly with the carrier frequency, and frequency hopping reduces the likelihood of interference [13], [102]. Also, a different ramp definition between the sensors with the same carrier frequency increases the likelihood that the signals cross each other [101].

The previous equations assumed a unit amplitude which does not apply in reality. We can adapt the power estimation in Equation 2.4 of a single target to a potential interferer. Therefore, we assume the same transmit signal power of the victim and the aggressor

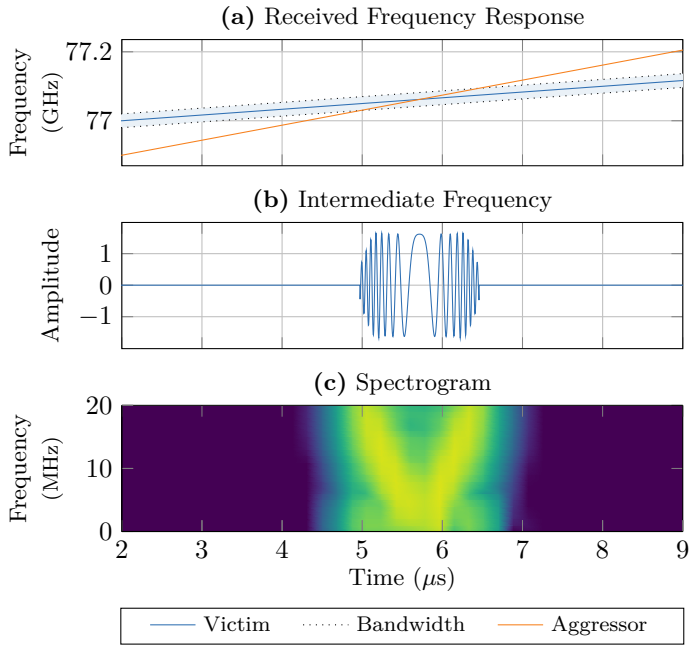

Figure 2.5: Simulated interference model: Analysis of the described interference model. (a) the instantaneous frequency aggressor crosses the bandwidth of the victim sensor. The low-pass filtered interference in (b) and the corresponding v-shaped spectrogram (c).

transmitter. A vehicle with an aggressor radar is a target and a potential interferer. Instead of the target-dependent properties, we change the equation to the direct signal of the interferer. Thus, the power P of an interferer is:

$$P_{interf} = \frac{P_{TX}^{a} \, G_{RX} \, G_{TX}^{a} \, \Lambda^2}{4\pi \, R^2} \tag{2.11}$$

where the receiver antenna gain is from the victim, and the transmit power depends on the aggressor sensor. The aggressor-specific variables are marked with an a for an aggressor. The relationship between distance and power leads to the following assumption. A target at the same distance will generate a lower signal amplitude than an aggressor sensor at a similar position [77]. Since an aggressor is always also a target, we know that the victim's signal needs twice the distance than the aggressor's signal. Therefore, the power of a single target will most likely be lower than the interferer.

However, the victim is only affected by interference signals within the receiver bandwidth. As mentioned, the receiver consists of multiple filters to prepare the signal for the ADC. Thus, we remove the signals outside the IF bandwidth with filters before we combine the interference and target signals. The combination of all interference signals

and target signals is a superposition of the individual signal components as defined by:

$$s_{\mathrm{RX}}[t] = \sum_{j=0}^{N_{\mathrm{targets}}} A_j \cdot \cos(\varphi_j[t]) + \sum_{i=0}^{N_{\mathrm{interf}}} A_i \cdot \cos(\varphi_i[t]), \tag{2.12}$$

with φ representing the instantaneous phase of the targets and interferer. The equation is the baseline for combining synthetic interference with real radar measurements used in Chapter 2.1.3.

Figure 2.5 demonstrates a simulated interference example scenario. The instantaneous frequency of the aggressor chirp is for a short duration within the victim frequency band. Since the interference frequency outside the bandwidth of the receiver filters does not influence the signal, we can neglect the signal outside the range of $5\,\mathrm{to}\,6.5\,\mu\mathrm{s}$. In (b), we demonstrate the corresponding time-domain signal with its strong amplitude. For the synthetic interference simulation, we use a low-pass filter to reduce the high-frequency components at the edge of the interference window to model the non-linear effects of the filter corner frequencies. The interference instantaneous frequency has at the edges $20\,MHz$ and in the center $0\,Hz$. The v-shaped frequency change over time is shown in Figure 2.5 (c). A regular target would lead to a constant frequency response in the spectrogram, while temporal linear frequency changes make interference visible.

2.2 Spiking Neural Networks

An SNN uses more biologically realistic neuron types than the ones implemented in ANNs. In contrast to the perceptron, the spiking neuron consists of a temporal memory and a non-differential spike function. In the following, we describe the essential components of an SNN, the basic neuron model, and two different variants used throughout this work. Afterward, we review the basic synaptic model and various connectivity schemes. At the end of this chapter, we will also investigate different SNN optimization methods, such as surrogate gradient and the biologically inspired Spike-Timing-Dependent Plasticity (STDP).

2.2.1 The Nervous System

A fundamental element of living being is the nervous system, a highly complex system responsible for receiving, processing, and transmitting information throughout the body. The name of the nervous system originates from nerve fibers, the axon of a neuron. In the following, we will describe the fundamental elements of the biological nervous system.

The nervous system comprises the peripheral and central nervous systems [103]. The peripheral nervous system connects the central nervous system to the rest of the body. At the same time, the central nervous system consists of the brain and the spinal cord. On the cellular level, the nervous system is composed of neurons. The neuron's cell body is called soma, which is responsible for the temporal integration of incoming action potentials and consists of the nucleus. The dendrites are the branching extension for

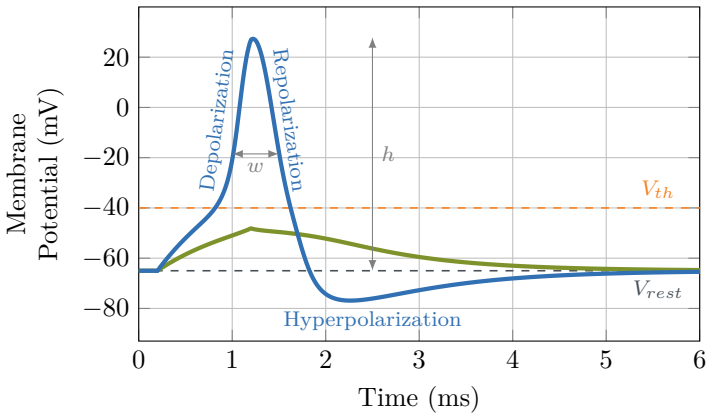

Figure 2.6: The action potential: The electrical impulse carries information between neurons via the axon, dendrites, and synapses. After exceeding the threshold, the action potential consists of a depolarization, repolarization, and hyperpolarization phase. The width w and the height h correlate with the parametrization of the neuron. We simulated the Hodgkin-Huxley neuron in the brian2 framework for the two cases of generating a spike and not reaching the threshold.

receiving signals from other neurons, while the axon is the fundamental element to send signals over a long distance to other neurons. To ensure a stable information transmission, the axon is isolated with myelin [104]. The synaptic cleft is often called a synapse between the axon and the dendrites. A rapid and dynamic transmission of signals across the tiny gate provided by the release of chemical neurotransmitters and their binding to receptors on [105].

Stereological studies estimate that each neuron has, on average, 7,000 synaptic connections, and approximately 20 billion cortical neurons exist [105]. The number of connections demonstrates the necessary computation complexity of the brain. Since the research on the biological nervous system exceeds this thesis, we refer interested readers to the publications [103], [105].

2.2.2 The Action Potential

The action potential, also called spike, is an event in time and is the fundamental element for information transfer in an SNN. Every neuron type uses a threshold to generate an output action potential, which we will describe in the neuron models section. A biological action potential is a short-term electrical response to stimuli forwarded across the axon to the synapses [106]. Figure 2.6 shows a simulated action potential based on biological observations. The subthreshold behavior switches to the spike response when the dynamics exceed the threshold. The spike has three phases: depolarization, repolarization, and hyperpolarization [107]. These phases are defined by the ion flow of sodium Na^+ and potassium K^+[106]. The hyperpolarization is also called the refractory

period since it reduces the likelihood of generating a new output spike with the undershoot potential.

So, the properties of an action potential vary as described by [108]. For example, the voltage peak is typically between 0 and $30\,mV$, and the width w is around 1 to $6\,ms$. [108] observed the variation of action potential and stated: "The spike shape depends on the availability of sodium channels, which depends on the voltage history, which in turn is correlated to the conductance input history." They have also proven that this variability does not correlate with noise and, therefore, has to carry additional information based on the neuron history.

In SNNs for engineering applications, we simplify the action potential to a unit amplitude and infinite small spike width because it reduces the algorithm's complexity. For the multiplication of synaptic weights, we can use simple memory weight lookups instead of complex multiplications [33]. However, using various spike shapes to allow more information is receiving more interest. In analog hardware, the shape variety can be modeled similarly to the biological counterpart. The digital implementation needs multiplications for the synaptic weights because the output spike varies in amplitude. The sparse and event-driven nature of the SNNs reduces the communication overhead between the neurons. Imagine a video frame consisting of 28x28 pixels with an average spike activity of 200 spikes and a maximum of one spike per pixel. This means only 25% of possible spikes transport information to deeper network layers, which reduces the necessary multiplications.

2.2.3 Neuron Models

The neuron is the essential component of all generations of NNs, and it always uses a non-linear activation function across the combination of input signals. The basic neuron utilizing the "all-or-none" functionality was introduced in 1943 by [109]. The McCulloch-Pitts neuron uses propositional logic to emulate neural events with a level detector that can represent any boolean function in a multilayer architecture. The second generation [14] uses non-linear activation functions like sigmoid or ReLU with continuous input values. The third generations use temporal spike events to process information with a Heaviside step function.

A spiking neuron consists of three components. The subthreshold dynamics describe the integration or resonating process of the incoming events or signals and the threshold, equivalent to the non-linear activation function, leads to a spike generation. The last component is the reset mechanism and the refractory period or hyperpolarization.

Throughout this work, we use 2nd generation neurons, which we will not explain in the background, and two different spiking neurons to build SNN architectures: the LIF and the R&F neuron. Many modifications exist in the neuron models, like the adaptive threshold or the Leaky-Integrator (LI) neuron.

Izhikevich has presented a comparison of the neuro-computational features of various neurons [110]. The LIF neuron is computationally efficient compared to the Hodgkin-Huxley and acts like a temporal integrator with a forgetting property. The LI is a LIF neuron without the firing mechanism, which makes it most suitable as an output

neuron. The R&F can use a resonating property with the unique feature of reaching the threshold through periodic excitatory and/or inhibitory spikes. This chapter describes the mathematical description and demonstrates the inherent properties. Additionally, we explain the Hodgkin-Huxley model since this is the first biological description that forms the basis for all subsequent neuron models.

Hodgkin-Huxley Model

In 1952, Hodgkin and Huxley discovered the first biologically plausible neuron model [111]. During experiments with the giant axon of a squid, they found the three main different ion currents: sodium, potassium, and a not further specified leak. Each ion channel transports a specific chemical element through the cell membrane, and the channel resistance can dynamically change depending on the opening or closing of the channel. Due to the different ion concentrations between the extra and intracellular and the electrochemical reactions, the neurons generate a Nernst potential [21]. The model is a four-dimensional differential equation system that describes the transport of ion current in an electrical circuit [21]. The current across the capacitor $I_c(t)$ is defined by:

$$I_c(t) = C\frac{u}{t} = -\sum_k I_k(t) + I(t), \tag{2.13}$$

where $\sum_k I_k(t)$ is the sum of all currents of the ion channels, and $I(t)$ is the input current. The membrane voltage is defined as u. Hodgkin and Huxley described a mathematical representation of the temporal change of three-channel resistances. The variable n controls the probability of an open K^+ gate, and the combination of m (opening) and h (blocking) defines the probability of the Na^+ gate. The three ion currents are given by:

$$\sum_k I_k(t) = \frac{1}{R_{Na}} m^3 h(v - v_{Na}) + \frac{1}{R_k} n^4 (v - v_k) + \frac{1}{R_L}(v - v_L), \tag{2.14}$$

with the reversal potential of the corresponding channel: v_{NA}, v_k, and v_L. The m, h, and n gates are given in the form of

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{1}{\tau_x(u)}[x - x_0(u)], \tag{2.15}$$

which defines the convergence of the x value to the resting value $x_0(u)$ with a time constant $\tau_x(u)$. The interaction between the input current and the different channels can lead to an explosive increase of the membrane potential, also called action potential or spike. Since the time constants of h and n react slower, the rapid increase is mitigated, and the membrane potential reaches the hyperpolarization phase. Figure 2.6 demonstrates a neuron with two different inputs for either generating a spike or staying in the subthreshold dynamics of the Hodgkin-Huxley (HH) neuron.

Leaky Integrate-and-Fire Neuron

The LIF neuron is the most common in various applications. Its first appearance dates back to 1907 [112] when Lapicque introduced an electrical circuit to investigate the firing frequency of nerve fibers. This IF neuron is computationally efficient because it reduces the action potential generation to a threshold function. Today, we still use the approximation of the action potential trajectory to reduce the computational cost compared to the HH model and take advantage of the more biological-inspired properties like the membrane potential leak or the refractory period.

The LIF neuron integrates the incoming current over time and emits an action potential if it exceeds the threshold. The membrane potential U of the LIF neuron i is defined by:

$$\tau_{\mathrm{m}} \frac{dU_i}{dt} = -(U_i - U_{rest}) + R \cdot I_i, \tag{2.16}$$

with the input resistance R and the input current I. The membrane time constant τ_{m} introduces a leaking effect such that U_i settles at the resting potential U_{rest} while no input activity is present. If the membrane potential exceeds the threshold ϑ, the neuron emits an action potential as defined:

$$\delta[t] = \begin{cases} 1, & U_i[t] > \vartheta \\ 0, & else, \end{cases} \tag{2.17}$$

where $\delta[t]$ is the Dirac delta function for the associated spike times t. In simulations, the amplitude of an action potential is simplified and set to one. The binary spike event combines the depolarization and repolarization phase, while the optional refractory period realizes the hyperpolarization. Thus, the neuron is either reset to a very low potential or deactivated for a certain duration.

For numerical simulation, a discrete form of the LIF neuron enables the usage of well-known deep-learning frameworks. Therefore, we can approximate the membrane potential $U_i[t]$ as:

$$U_i[t+1] = \underbrace{\alpha \cdot U_i[t]}_{\text{leak}} + \underbrace{R \cdot I[t]}_{\text{input}} - \underbrace{U_{\text{reset}} \cdot \delta(t)}_{\text{reset}}, \tag{2.18}$$

with the leak factor α defined by $exp(\frac{\Delta t}{\tau_{\mathrm{m}}})$, where Δt is the simulation time step size and the membrane time constant τ_{m}. The input resistance and input current define the input potential from the synaptic connections, and during a spike generation, the membrane potential is reset. As mentioned, the reset is sometimes implemented by a hyperpolarization or a refractory period, adding additional costs to the simulation. The neuron activation utilizes the Heaviside step function to emit an action potential:

$$\delta(t) = \Theta(U_i[t] - \vartheta), \tag{2.19}$$

with the threshold potential ϑ and the Heaviside function Θ.

A variant of the LIF neuron is the LI neuron. It models the same dynamic except for the Heaviside step function and the refractory period. Such a neuron acts as a pure

integrator and provides the possibility to count spikes with a leaking effect. The LI neuron is often used as a decoding neuron in the output layer [113], [114].

Resonate-and-Fire Neuron

The R&F neuron proposed by [115] is a symmetric two-dimensional model with oscillatory behavior. The dynamics of the subthreshold behavior are described by

$$\begin{aligned} x' &= bx - \omega y \\ y' &= \omega x + by, \end{aligned} \tag{2.20}$$

where b is the damping constant, ω is the neuron's resonance frequency defined as $2\pi f_0$. The variables x and y represent the current- and voltage-like variables. The equivalent complex form of the linear system is

$$z' = (b + \mathrm{j}w)z, \tag{2.21}$$

with $z = x + iy$. The general solution of the differential equation is

$$z\,(t) = C_1\,\mathrm{e}^{t\,(b+\mathrm{j}w)}, \tag{2.22}$$

with the assumption that C_1 is a constant unit factor. For the simulation of the neuron, it is possible to use exact solutions of the differential equation, but this limits the usage. The discretization provides higher flexibility to build the neuron into existing NN frameworks. Hence, we use the exact integration [116] to define the discrete representation of an R&F neuron:

$$z_k[n] = \underbrace{\mathrm{e}^{\Delta t\, b_k}}_{=\lambda}\,\mathrm{e}^{\mathrm{j}\,\Delta t\,\omega_k}\,z[n-1] + a_k[n], \tag{2.23}$$

with input current $a_k[n]$ of the neuron with index k and the step size Δt. The equation for one R&F neuron with index k consists of two components, the damping factor λ and the oscillation kernel $\mathrm{e}^{\mathrm{j}\,\Delta t\,w_k}$. The damping factor leads to an exponential decay since $b < 0$; therefore, λ is defined in [0,1]. The oscillation frequency depends on the individual neuron's resonance frequency f_k and defines the phase shift per time step. The discrete equation can also be defined by using the Euler formula:

$$\begin{aligned} u_k[n] &= \lambda(u[n-1]\,cos(\theta) - v_k[n-1]\,sin(\theta)) + a_k[n], \\ v_k[n] &= \lambda(u[n-1]\,sin(\theta) + v_k[n-1]\,cos(\theta)), \end{aligned} \tag{2.24}$$

where u and v represent the real and imaginary components, respectively. θ corresponds to the exponent of the oscillation kernel $\Delta t\,\omega_k$. We use Equation 2.24 in our simulations because PyTorch [117] currently does not fully support arithmetic operations with complex tensors and has limited compatibility with Just-In-Time (JIT) compilation.

The generation of an output potential is similar to the LIF when the voltage-like variable $\mathrm{Im}(z)$ exceeds the firing threshold ϑ, the neuron generates an action potential, and the internal states are reset. In the original publication of Izhikevich [115], the

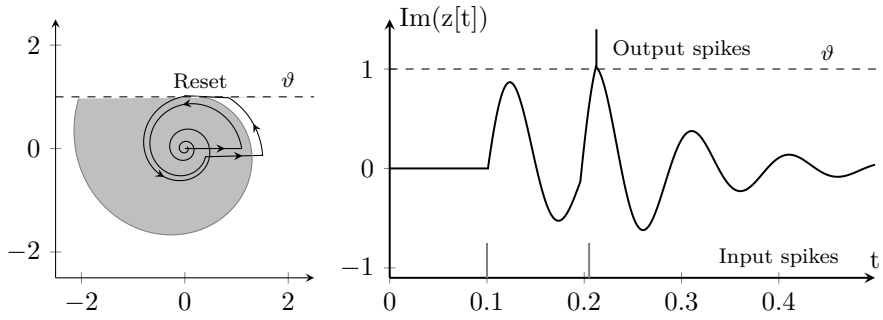

Figure 2.7: R&F neuron with spike activation. Input spikes with the same frequency as the resonance frequency of the neuron lead to an exceeding of the threshold and an output potential. When the state exceeds the gray area of the phase portrait, then always an output spike will be generated. Simulation is based on the discrete R&F neuron.

current-like variable is reset to zero. Therefore, the neuron continuous the damped oscillation because the imaginary component is unaffected, as shown in Figure 2.7. This is different in [43], where the real and imaginary component is reset to zero and further oscillation is suppressed. This reduces the likelihood of an output action potential because the excitation of a neuron starts from the beginning. After a single spike input, the imaginary components are shifted from the real part by 90°, enabling the oscillation.

There are also different variants of utilizing the threshold functions. In [43], an adaptive threshold is used to reduce the number of spikes for strong input stimuli. Simultaneously to [118], we developed the phase-dependent spike mechanism to encode the phase of a signal in the spike timing. Thus, the threshold of the real part is combined with the requirement that the imaginary component equals zero. For implementation in a time-raster simulation, we approximate the requirement by detecting the sign switch of the current and last imaginary component. Additionally, [118] combined the R&F phase-dependent spike mechanism with graded spikes, meaning the spike has an amplitude. This idea has some fundamental advantages but increases the computational complexity of the subsequent synapses.

Since the neuron is complex, it is possible to input and output complex spikes. A complex input makes sense if the preceding layer provides complex information. However, the complex output, in combination with the phase-dependent spike mechanism, does not carry any additional information since the imaginary component is zero when the spike appears. Thus, we implement our R&F neuron only with a real spike output but an optional complex input.

An incoming action potential affects the real component of Equation 2.23. Considering two positive input spikes with a distance equal to the period of the resonance frequency of the R&F neuron, then the amplitude of the oscillation will increase, as demonstrated in Figure 2.7. Here we use the original spike mechanism of [115]. If the second spike appears at a half period, the amplitude of the oscillation would be decreased; the oscillation would not reach the threshold, demonstrating frequency selectivity. With the phase-dependent

Table 2.1: Neuron model complexity: Computational complexity with the number of operations of each neuron subthreshold dynamics for each time-step n. Therefore, the various spike and reset mechanism variants can be used for all biologically inspired neurons and are neglected here. The HH operations are extracted from the implementation of [123].

Neuron Model	Multiplications	Additions
HH	25	21
LI	2	1
LIF	3	2
R&F	6	3

spike mechanism, we ensure that the spike always appears at the highest peak of the oscillation. With the idea of the graded spikes, the amplitude information of the signal can be recovered.

The properties of an R&F neuron are mainly defined by the damped harmonic oscillation that carries historical stimuli information of the specific neuron. A population of R&F neurons can use the resonating property to act as memory, as demonstrated by [119]. Similar behavior was observed in biological neurons. The balance of excitatory and inhibitory synapses with recurrent connections can act as oscillators.

Summary of Neuron Models

In the neuron models shown so far, there is a considerable variation of different types with various dynamics, like the spike response model [120], the Galves-Löcherbach model [121], the FitzHugh-Nagumo model [122], and many more. Izhikevich compared selected neuron models regarding their biological plausibility and the implementation costs in FLOPS [110]. Biological properties are spike bursting, spike frequency adaptation, hyperpolarization, and more. Thus, a neuron that supports more biological features requires a complex model with a higher computational cost.

Similar to [110], we summarize the number of multiplications and additions in Table 2.1. Note that we assume a digital neuron implementation without considering the synaptic connection. We assumed the neurons are implemented for a time-raster simulation where each neuron consumes the shown multiplications and additions per time step. However, event-driven implementations are less predictable in terms of their computational complexity due to the dependency on the number of input events. Also, the neuron does not need to be updated continuously when no inputs occur, as demonstrated by [124], [125].

In addition to the computational complexity difference, the neurons use fundamentally different biological features. While the LIF neuron integrates incoming spike events, the R&F resonates, and the exact timing of the input spikes affects the neuron states. Both neurons can be used for continuous input data because they allow the temporal information to leak. One significant difference is the effect of inhibitory spikes, meaning a negative spike potential. In the LIF, the membrane potential reduces during a negative

input, while in the R&F neuron, it depends on the exact timing of the spike. A spike can either decrease or increase the oscillation's amplitude depending on the current position of the oscillation, either at the lowest or highest peak. The LI neuron is similar to the LIF except for the spike function, providing an analog output value, and is mainly used as an output layer.

2.2.4 Synaptic Connections

The synaptic cleft is a specialized junction connecting neurons and enabling communication. The electrical pulses from pre-synaptic neurons are transported across the ≈ 20 nm wide cleft by electrochemical reactions [126]. The release of neurotransmitters into the synaptic cleft results from a pre-synaptic activation. The transmitters move to the other side by diffusion and activate the receptors [21]. The synaptic vesicle stores various transmitters with a limited capacity [127]. The diffusion of neurotransmitters leads to a structural change of the ionotropic receptors, which convert the chemical reactions to electrical signals. The duration, direction, and number of ions within the ion channel affect the gain of the post-synaptic potential [128]. Receptors with inward current lead to a depolarization of the membrane potential, called Excitatory Postsynaptic Potential (EPSP). Receptors with outgoing current discharge the membrane potential by Inhibitory Postsynaptic Potential (IPSP). Metabotropic receptors directly trigger intracellular events, such as opening an ion channel [129]. In contrast to the chemical synapses, the electrical has a gap width of ≈ 3.8 nm and is bidirectional [130]. Both synapses can either act inhibitory or excitatory.

Due to the high complexity of the biological synaptic connections, mathematical models imitate the synaptic behavior. These simplifications neglect some aspects that could be important for understanding the brain but less relevant in neuromorphic computing. This section focuses on the synaptic connection, especially the static and dynamic synapse model and the common connection schemes.

Static Synapses

The static synapse is the simplest possible synaptic connection because it reduces the connection to two main parameters: weight and delay. The weight defines the direction and strength of the current flow from the pre-synaptic to the post-synaptic neuron. This synapse also models the charging processes of the transmission by a decay β, which mainly uses an exponential or alpha-shaped function. We use the following discrete expression:

$$I_j[t] = \beta I_j[t-1] + \sum_j \boldsymbol{W}_{ij}\delta_i[t], \tag{2.25}$$

with the synaptic connection between neuron i and j. The exponential decay is defined by $\beta = exp(\frac{\Delta t}{\tau_{\text{syn}}})$, with the synaptic time constant τ_{syn}, which can deviate between inhibitory and excitatory connections and the simulation time step size Δt. Also, in biological systems a propagation delay exists due to transmission speed, distance, and

neurotransmitter release [131]. In event-driven simulators, it is possible to model the synaptic delay, and in time-raster simulations, the delay is implemented as a temporal shift.

Dynamic Synapses

In contrast to the static synapse, the dynamic changes its strength or weight over time depending on the historical input. This mechanism regulates the information flow within a neural circuit and balances the excitation and inhibition through short-term depression and facilitation.

Short-term depression refers to decreased synaptic strength during repeated activation from the pre-synaptic neuron [132]. Thus, the previous activation consumes the available neurotransmitters and weakens the amplitude of the post-synaptic response until the neurotransmitters have recovered, explaining the naming short-term.

On the other hand, short-term facilitation increases synaptic strength due to repeated activation temporally [133]. Due to the constant activation, a residual calcium ion channel appears, and leads an increase of the synaptic strength. With no activation the facilitation recovers after a short period.

The long-term depression and potentiation [134] are very similar to their short-term counterpart, except for the timescale of effect and recovery. Short-term refers to a timescale of seconds, while long-term could last for hours, days, or even longer [135].

The Tsodyks-Markram model [136] describes the dynamics of short-term plasticity by a set of differential equations that account for neurotransmitters' release, depletion, and recovery. A high pre-synaptic activity results in a rapid depletion and a following short-term depression, while a low activity leads to facilitation. The well-studied short-term plasticity model exists in many variants [137], [138]. However, in this work, we use mainly the static synapse model because different dynamics increase the computational complexity.

Connectivity

Besides the synaptic model, the connectivity of spiking neurons differ from the second generation networks. Traditionally, there are only two connections schemes: feedforward, and recurrent. Feedforward describes the connection of consecutive neuron layers from the input to the output. Thus, the post-synaptic neuron depends only on the pre-synaptic input. Recurrent networks use a bidirectional information flow, meaning that the output of a neuron can be connected to the previous layer. Thus, we define the synaptic input to a single neuron by:

$$I[t] = \underbrace{\sum_{i=0}^{N_{\text{ff}}} \boldsymbol{W}_{\text{ff}_i} x[t]}_{\text{feedforward}} + \underbrace{\sum_{j=0}^{N_{\text{rec}}} \boldsymbol{W}_{\text{rec}_j} I_l[t-1]}_{\text{recurrent}}, \tag{2.26}$$

where I_l is the previous output of the neuron l that can be located deeper in the network. For a feedforward network this equation simplifies to the first term that further reduces the training complexity. Equation 2.26 is valid for traditional and spiking networks when considering that the input I_l and x can be spikes. The weight polarity decides whether the spike or the input data are excitatory or inhibitory. It is worth to note that convolution layers are just a variant of the feedforward connected architectures with intensive weight sharing. Thus, every convolutional layer can be converted to a feedforward representation and be applicable to spiking networks, as demonstrated by [23], [139].

However, in spiking networks, we are not limited to think in layers. Thus, we like the idea of a neuron population where the individual elements can be placed in a three-dimensional environment. One extensive example is the concept of the reservoir, meaning a population of randomly connected neurons convert the low-dimension input into a non-linear high-dimension representation. A more understandable example is a bucket of water with the input of a stone. The stone stimulates water waves that can be interpreted by cameras or detectors [140]. The reservoir is called echo state machine for traditional networks [141] and liquid state machine for spiking networks [142], [143].

2.2.5 Training Algorithm

The learning procedure represents one of the most important aspects of neural systems since every living being learns new from experiences and applies the pre-learned knowledge to various unseen tasks. The process of learning involves structural changes in the nervous system architecture as well as modification of the morphology of the individual neurons [130]. In traditional networks, learning is limited to the synaptic weights, and the architecture is in the hand of the developer defining the hyperparameters. In spiking networks, we can additionally train the synaptic delay, time constants, neuron threshold, and more depending on the neuron type.

There exist a variety of training methods for spiking networks. Compared to ANNs, there is not yet a single method that utilizes all the dynamical features of these neurons. Thus, we can define the two conversion categories and direct-based training methods. The conversion method relies on a previously trained ANN where the signals are converted to a rate or Time-To-First-Spike (TTFS) spike representation [144], [145]. This is possible since the ANNs are a rate-based approximation of the biological neuron.

The direct training method ranges from biologically inspired synaptic plasticity to neuroevolution and gradient descent. In this thesis, we use the gradient descent method to train SNNs. However, we provide a short review of synaptic plasticity and neuroevolution in the following.

Previously we defined the difference between short-term and long-term plasticity while explaining the dynamic synapse. Short-term plasticity reacts on a short timescale and is a technique to balance the population activity. For example, the auditory system's frequency adaptation is a short-term adaptation process for signal amplitude changes [146]. Learning long-term plasticity is crucial, like in STDP. It is inspired by the Hebbian learning rule, which states: "When an axon of cell A is near enough to excite cell B

and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" [147]. In simpler words it means "neurons that fire together, wire together" [148]. Thus, the synaptic weight increases for two connected neurons when the pre-synaptic neuron fires shortly before the post-synaptic neuron. The weights decrease if the pre-synaptic neurons fire after the post-synaptic neuron. For more details about STDP, we refer to [21], [149]. The original idea of STDP follows an unsupervised learning method, but there are also supervised variants [150].

Evolutionary algorithms are very promising because of the simultaneous optimization of the network architectures and parameters [151]. This algorithm follows the idea of evolution, meaning the survival of the fitters. Thus, the algorithm involves the following steps: 1) the random network definitions and 2) utilizing a fitness function to evaluate the performance of each network. 3) selecting the best network used for 4) generating a new group of networks via random generation, crossover, and mutation. The crossover combines the parameters of two-parent networks while the mutation introduces random changes to the best-performing networks. To retrieve a small network architecture, it is possible to include in the fitness function a network size penalty. Evolutionary training is a smart parameter search inspired by nature, and [26], [152] has shown well-performing small networks. Besides the capabilities of optimizing with hardware constraints, the process of evolution is time and computationally expensive and is less suitable for large networks and datasets.

The conversion-based training methods cannot take advantage of the neuron dynamics, and the network architectures are limited to the ANN feature space. On the other hand, biological methods like STDP and evolution take advantage of the neuron dynamics but are either unstable or computationally expensive. So, one idea is using gradient descent training with spiking neurons, which we use throughout the work. In the following, we explain the theory and background of surrogate gradient learning.

Backpropagation

Gradient optimization is a method to find the minimum or maximum of a function. The gradient of a loss function concerning the weights of a network for a single output is a part of the backpropagation algorithm. So, the gradient for each layer is computed once and iterated backward through the model. Backpropagation is commonly combined with gradient descent or stochastic gradient descent to update the weights of the individual layers [153].

To use gradient optimization, the following assumptions need to be valid: 1) the function is differentiable with respect to its parameters, 2) the function's derivative exists and is continuous, and 3) the function is convex with unique minima or maxima. However, in reality, the function $f(x)$ consists of local minima or maxima, and gradient descent tries to find the global minima by analyzing the directional derivative of $f(x)$. The vector of all partial derivatives $\frac{\delta}{\delta x_i} f(\boldsymbol{x})$ is the gradient of f denoted as $\nabla_x f\boldsymbol{x}$ [17].

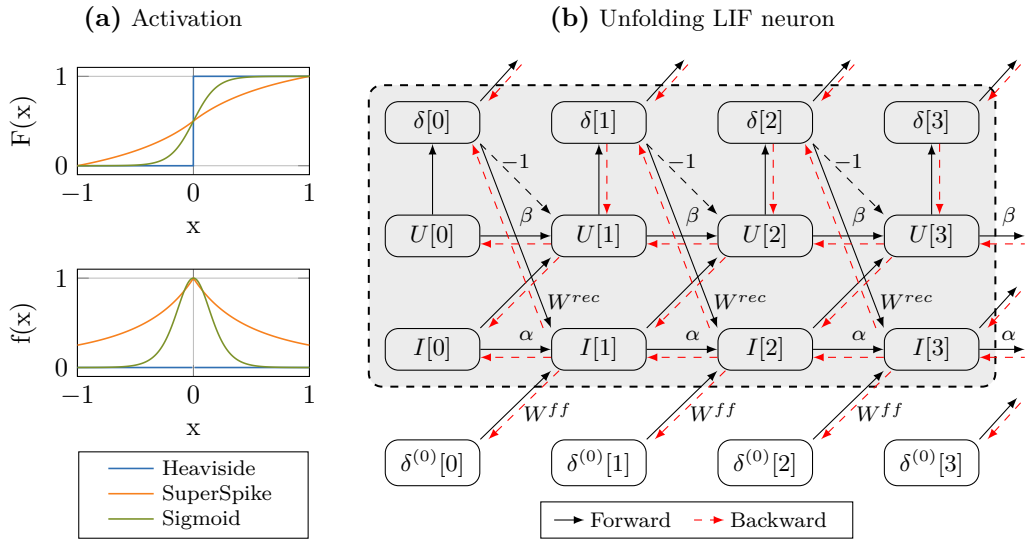

Figure 2.8: Spiking backpropagation: (a) approximation of the Heaviside step function of the neuron activation with superspike and sigmoid. The first plot demonstrates the function, and the second the corresponding derivative. The derivate of the Heaviside is not defined for zero. (b) demonstrates the temporal unfolding of a neuron for backpropagation through time learning. The input can either be a previous layer or direct signals. The gray area corresponds to a single LIF neuron.

The iterative calculation of the next position follows the equation:

$$x' = \boldsymbol{x} - \eta \nabla_x f(\boldsymbol{x}), \tag{2.27}$$

with the learning rate η, scaling the step size. In a multiplayer network, the error must be propagated backward across each layer by following the chain rule [17].

Every spiking neuron has a threshold function that generates an output action potential when the neuron state exceeds a threshold. This threshold is a Heaviside step function which is either zero or one. The derivative of the Heaviside step function $H(x)$ is undefined for $x = 0$ and otherwise zero. Thus, $H(x)$ is not compatible with the assumption 2) of the gradient optimization. To overcome this issue, the idea of surrogate gradients uses an approximation function for the backward path. In Figure 2.8 (a), we demonstrate the approximation of the Heaviside step function with its derivative. In contrast to the original function, the approximated derivatives are continuous. The upper plot demonstrates the functions, and the lower demonstrates their derivatives. Each approximation function has a tunable parameter which we call scale since it is a smoothness or steepness factor, meaning a higher scale better approximates the Heaviside step function. From an implementation perspective, the forward path executes the Heaviside step function and the backward the derivative of the approximation function called surrogate gradient. [38] has demonstrated that the surrogate gradient is very robust, and the selected surrogate gradient is less important for the final network performance, while the derivative's scale

highly affects the training performance. The superspike [154] derivate is defined as:

$$f'(x) = \frac{1}{(\alpha|x| + 1.0)^2}, \tag{2.28}$$

where α is the scaling factor, and the sigmoid derivative is:

$$f'(x) = \sigma(x)\,(1 - \sigma(x)) \text{ with} \tag{2.29}$$

$$\sigma(x) = \frac{1}{1 + e^{-\beta x}}, \tag{2.30}$$

where the scaling factor is β. Since the computational complexity is much higher for the sigmoid, we selected for all our networks the superspike approximation of the activation function with $\alpha = 100$.

Because a single spiking neuron has an inherent recurrency by its temporal-dependent membrane potential, the gradients must be backpropagated through time. Thus, the computational graph needs to be unfolded in time [155], meaning that a simulation of 128 times steps for a single neuron increases the computation graph from one to 128 nodes. Back-Propagation Through Time (BPTT) is a well-known algorithm in traditional NN, and it was first presented by Werbos in the 90s [156]. We demonstrate the general idea in Figure 2.8 (b), where we use a single LIF neuron that we unfold over time (gray area). The input to the spiking neuron can either be a spike or a sampled input signal. The input contributes to the synaptic state depicted as $I[t]$ with the forward weights W_{ff}, shared across the time dimension. The synapse directly contributes to the membrane potential that depends on the previous state. Thus, all states are randomly initialized at $t = 0$. The threshold of the membrane potential decides if an output spike is generated and triggers the reset of the membrane potential. The output can then be connected to a further layer or the loss function for the error calculation.

Also, we demonstrate in Figure 2.8 (b) the backward path. In general, the error is back propagated until the input of the layer except for the reset mechanism. We detach the reset from the computation graph because it is not differentiable. There are methods like soft reset to make the reset function differentiable. However, we neglected this part to reduce the computational complexity during training. Additionally, there are known problems with BPTT and surrogate gradient besides the high memory consumption:

- **Local minima:** The local minima problem states that the training algorithm is stuck in local minima and cannot reach the global minima [157]. Advanced optimizations like momentum and adaptive learning rate improve gradient optimization.

- **Vanishing gradient:** In BPTT, the computational graph explodes with the time dimension and thus is susceptible to gradients close to zero [158]. To prevent the network does not train due to vanishing gradients, it is common to use a truncated BPTT [159].

- **Dead neuron problem:** Since the spiking neurons only emit zeros and ones, the network is pruned to dead neurons. It means the neuron output is permanent zero, and the weights will never receive an update during back-propagation. It permanently damages the neuron because the gradient cannot flow to the previous layers, resulting in an untrainable network [160]. The dead neuron problem can occur for different reasons: 1) the weight initialization with zeros and 2) big negative weights for integrating neurons.

Sparsity Regularization

The previously described training methods can include a term that increases the computation efficiency of the algorithm. On the one hand, reducing the number of operations, the complexity, or the number of bits (digital) improves the power consumption. One recent development is the reduction of the 32 floating point to an 8-bit floating point representation without performance degradation [161]. However, there are other methods to improve the efficiency of the trained networks, especially within the domain of SNNs. The number of spike events directly correlates with the power consumption, and the number of memory accesses increases the energy footprint due to the complexity of the architecture [33]. Also, the processing time of event-driven implementations is sensitive to increasing events [162]. Therefore, we describe here two methods to reduce the spike count and the number of synaptic connections that apply to the surrogate gradient and are used in this work.

Spike Activity Regularization:
Spike regularization is a technique to force the network to reduce the number of synaptic events that correlate with the power consumption of the network depending on the neuromorphic implementation. Therefore, we extend the loss function by a spike activity regularization term to encourage the training to reduce the number of spike events. In this work, we utilize the average spike activity of the individual sum of spikes across time:

$$\mathcal{L}_{\text{spike}} = \lambda \frac{1}{NB} \sum_{b=0}^{B} \sum_{n=0}^{N} \sum_{t=0}^{T} \delta_n^b[t], \tag{2.31}$$

where T denotes the temporal length of the input data, N is the number of neurons in the given layer, B represents the batch size, and λ scales the impact of the spike activity to the total loss. Therefore, the network reduces the number of spike events during training. For high λ, the training could end up in zero spikes because the rewards for no spike is higher than for a good classification. A solution to prevent the dead neuron problem was proposed by [38], where they use an upper and lower bound to reduce the likelihood of no spike events. Another method uses the quadratic Euclidean distance of spike events [163]. In general, it is a good estimate to use a λ which reduces the mean spike activity into an equivalent range as the expected loss for the given dataset. It is important not to mix up the spike regularization with the spike frequency adaptation, which mainly appears in higher biological-inspired dynamical systems like the threshold adaptation

[164]. The difference is the target of the techniques to reduce the spike rate or to adapt on a short-term scale the firing frequency.

Synaptic Sparsity:
Another way to increase sparsity is by reducing the number of neurons or synaptic connections within the spatial domain. The network architecture can be optimized by a parameter search or during an evolutionary optimization. Another technique is described by the Lottery Ticket Hypothesis: *"A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations"* [165]. It basically states the existence of an isolated subnetwork that performs similarly to the original network with more parameters. In [165], the focus was on the experimental proof of the hypothesis, where they retrained the best network after the pruning, called a winning ticket. Therefore, they could reduce the network size to $10 - 20\%$ of the initial network because the initial weights make the training of the winning ticket effective. They proposed an iterative pruning method to constantly remove the weights with the lowest magnitude during the training.

Biological-inspired SNNs often use a sparse initialization, demonstrated by reservoir computing, where the network is randomly initialized, and only the connections to the readout layer are trained. The sparse connectivity is necessary to reduce the likelihood of continuous firing due to the recurrence.

The Deep Rewiring algorithm [166] is another gradient-supported method to remove and generate connections. Therefore, each synaptic connection consists of a constant sign (excitatory or inhibitory) and a trainable parameter ϑ. If the parameter ϑ is positive, the connection is active; otherwise, it is obsolete. A new randomly selected synapse is generated based on a uniform distribution for each removed synapse. Another rewiring method was proposed by [163], which uses the magnitude of the absolute weight as a pruning score. The momentum derived from the exponential temporal moving average of the gradient and the spatial distance defines the probability of synaptic growth. Unlike the rewiring algorithms, we use a less complex pruning algorithm without connection growth. After the selected network is trained, the weights with a magnitude close to zero are randomly removed depending on the expected pruning rate. A relationship between the number of removed and total connections defines the pruning rate. Whenever the loss increases, retraining, also called fine-tuning, improves the final performance [167]. This pruning algorithm is sometimes repeated iteratively to perform a scheduled pruning for better final performance.

3 Information Encoding

Information encoding refers to the process of transforming information into a streamlined, concise, efficient, and easily comprehensible form for a given system. It serves as an interface between two systems, similar to how the ASCII code translates bits into human-readable representations. Conversely, decoding involves converting letters and numbers back into their binary representations, thus reversing the encoding process.

In biological systems like the brain, discrete electrical impulses transport information between neurons across synaptic connections, as explained in Chapter 2.2. Two studies discovered that various encodings represent data from different biological sensor data like visual, acoustic, control, or somatic data [168], [169]. Therefore, the application-specific encoding of the biological model suggests that an efficient encoding scheme depends highly on the application. However, the encoding type greatly influences the number of spikes and correlates with the power consumption of event-based neuromorphic hardware [170].

There is information that follows a specific time grid, like images or clock-driven systems. The image is recorded at constant intervals, and each pixel consists of at least one floating point number for mono-color code. These values can then be converted into a specific spike pattern and processed in an SNN [171], where the time between the spike event and the reference point (start of the image processing) carries the floating point value. Hypothetically, a simulation with infinite small-time steps also has an infinite small resolution of the pixel value, which is very unlikely to achieve due to hardware limitations. However, "*the duration of the encoding window used for any analysis of the neural code is not arbitrary but depends on the dynamical nature of the information being encoded*" [172]. This means the time between the input images defines the maximum representable number.

Another type of information are continuously changing signals, which could be sensor data from temperature, ultrasonic, radar, and more. These sequential signals can have frequencies in the range of MHz, making a good resolution with spikes challenging. One way to overcome this resolution limitation is converting the sequence into a frame-based representation through preprocessing with feature-extracting methods. With such an encoding, we reduce the temporal advantages of SNNs. Thus, we highlight the capabilities of the encoding to convert frame or stream-based data.

There is not yet the answer to which encoding scheme to use. Thus, we provide an overview of the different encoding methods based on our publication [39] and extend it by the inherent properties of the R&F neuron. In the application Chapters 4, 5, and 6, we will compare selected encoding schemes and identify a suitable representation for the sinusoidal signals.

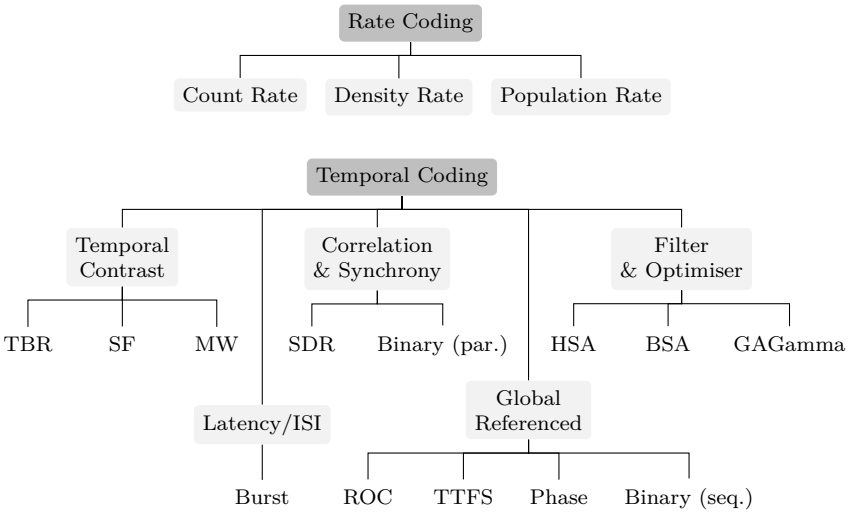

Figure 3.1: Encoding taxonomy: Overview of different encodings separated into rate and temporal techniques. Previously published in [39].

3.1 Taxonomy and Biological Inspiration

We categorized the current coding schemes into two distinct clusters: rate and temporal codes [39]. The main difference between these two variants is whether the number of spikes or the exact timing carries the information. The rate codes use the instantaneous or average spike rate of the neurons. In contrast, the temporal schemes use the exact timing of a spike by interpreting the spike time in correlation to a fixed reference, by order of occurrences, or with the interval between spikes. Consequently, a jitter of the spike timing impacts the temporal schemes but does not influence the rate codes. Often there is the additional category of population codes, which we omitted since the individual encoding of the neuron follows either the rate or the temporal scheme. A population is defined as a group of neurons in a two or three dimensional space. Figure 3.1 represents our taxonomy of different encoding schemes.

Rate codes are the base for the second-generation networks, which also makes the conversion between ANNs and SNNs possible [173]. Since neurosciences proposed that rate coding is the default information transportation technique of the nervous system [169], it is still predominant in the current research of SNNs [174]–[176]. After Thorpe suggested that information can be encoded in the exact spike timings [177], neuroscientists found experimental evidence: for the processing of an object classification task, the human visual system needs only 150 ms which cannot be explained by rate codes [178] and further experiments promoted these findings for visual [168], [179]–[181], audio [182], tactile [183], and olfactory system [184], [185]. Another experiment showed that a mouse could discriminate between odors within 200 ms, which can take 100 ms longer for similar smells [183].

Biological evidence showed no single encoding scheme for all sensor elements, which we also observed in the application-oriented community of SNNs. So, applications with fast-changing input rely on temporal codes, while slow frame-based data can represent their information in rates. Unfortunately, there is no universal solution to the question of the most appropriate coding scheme.

3.2 Rate Coding

The general idea of rate coding represents information within the spike activity over time, neurons, or simulations. One general known drawback of rate encoding schemes is the generation of many spike events, making the event-based execution slow and inefficient [33].

Count Rate *(average over time)* is the average spike events across time of the individual neurons. The count rate r of neuron n is defined by:

$$r_n = \frac{1}{T} \sum_{t=0}^{T} \delta_n[t], \tag{3.1}$$

where T is the time length of the observation window and $\delta[t]$ are the spike events at time t. Another name for these schemes is frequency coding [186]. Often the spike times are modeled exactly, by a linear distribution of events, or random, by the Poisson distribution. The count rate encoding was observed by Adrian and Zottermann by the change of the firing rate due to the stretching of a frog muscle with different weights [169].

Density Rate *(average over runs)* estimates the rate across the neural activity across multiple runs. Thus, the system receives the same input for multiple stimulations meaning that such an encoding is not real-time capable. Imagine a frog trying to catch a fly with density rate coding. It would mean the fly has to use the same trajectory multiple times so that the frog can catch the fly, demonstrating that density rate is only an encoding in ideal environments [21].

Population Rate *(average over neurons)* represents the average spike activity across a neuron population within a time interval. Population rate is defined by

$$r_t[t : t + \Delta t] = \frac{1}{N \Delta t} \sum_{n=0}^{N} \delta_n[t : t + \Delta t] \tag{3.2}$$

in the time interval starting at t with the window size of Δt. The neurons N generate synchronized spikes, which reduces the computational efficiency of the system. Each neuron of the population is responsive to a specific input, as it is often described as tuning curves. The superposition of the neuron activity carries the information about the input value [173].

The usage of the inherent properties of the LIF neuron as an encoding stage is called current injection because the input current is directly connected to the input synapses of the population. The input weights can be modified such that each individual neuron has

a different firing frequency for the input range. Another extension is the receptive field, mainly based on Gaussian kernels and inspired by the human's visual encoding [187].

3.3 Temporal Coding

The second category of encoding schemes is temporal codes which use local or global references to the spike timing. Methods where the exact timing is less important but the order of the spike events from different neurons still fit into this category. Thus, there are many subcategories of the temporal schemes. The Temporal Contrast (TC) utilizes the signal's derivative and focuses on the amplitude changes. The Latency and Global Referenced use a global or local reference of the spike timing. The similarity between the Inter-Spike-Interval (ISI) encoding and correlation & synchrony schemes exists since the information is encoded in the difference between spikes by single or multiple neurons. The last schemes fit the filter-based category because they use a kernel function to extract the spike patterns. In Figure 3.2, we provide an overview of the temporal encoding methods.

In the following, we describe some selected methods shown in the taxonomy in Figure 3.1, which are important for this work. Our publication [39] provides a full overview of the encoding techniques.

3.3.1 Global Referenced

The most common temporal encoding scheme is TTFS, where the information is encoded in the distance between a global reference and the spike. It means an earlier spike has a higher amplitude while low amplitudes have a large interval. Therefore, the spike timing is the inverse of the stimulus amplitude a defined as $\delta t = 1 - 1/a$. The importance of the relative time of the first spike was discovered during experiments with discrete mechanical fingertip events [183]. Likewise, observation from the retinal pathway strengthens the usage of the time information by demonstrating the invariant relation of the stimulus contrast [168].

Another version of the referenced encoding scheme uses an oscillating reference point called phase coding. Thus, the time between the spike and its closest oscillation reference point carries the information [188], [189]. It extends the static idea of the TTFS encoding to continuous data. The phase coding was observed in the cat's visual cortex by identifying the spike pattern and the reference oscillation [190].

While TTFS coding uses the exact timing, Rank-Order Coding (ROC) uses the spike order of a neuron population concerning the global reference point. Thus, the exact amplitude information is lost with the benefit of resistance to jitter and noise. Often in globally referenced coding schemes, the first spike is the most important, enabling the processing in an importance order. Therefore, the longer the network processes incoming spikes, the more accurate it will be, demonstrating the speed-accuracy trade-off of global referenced coding schemes [191].

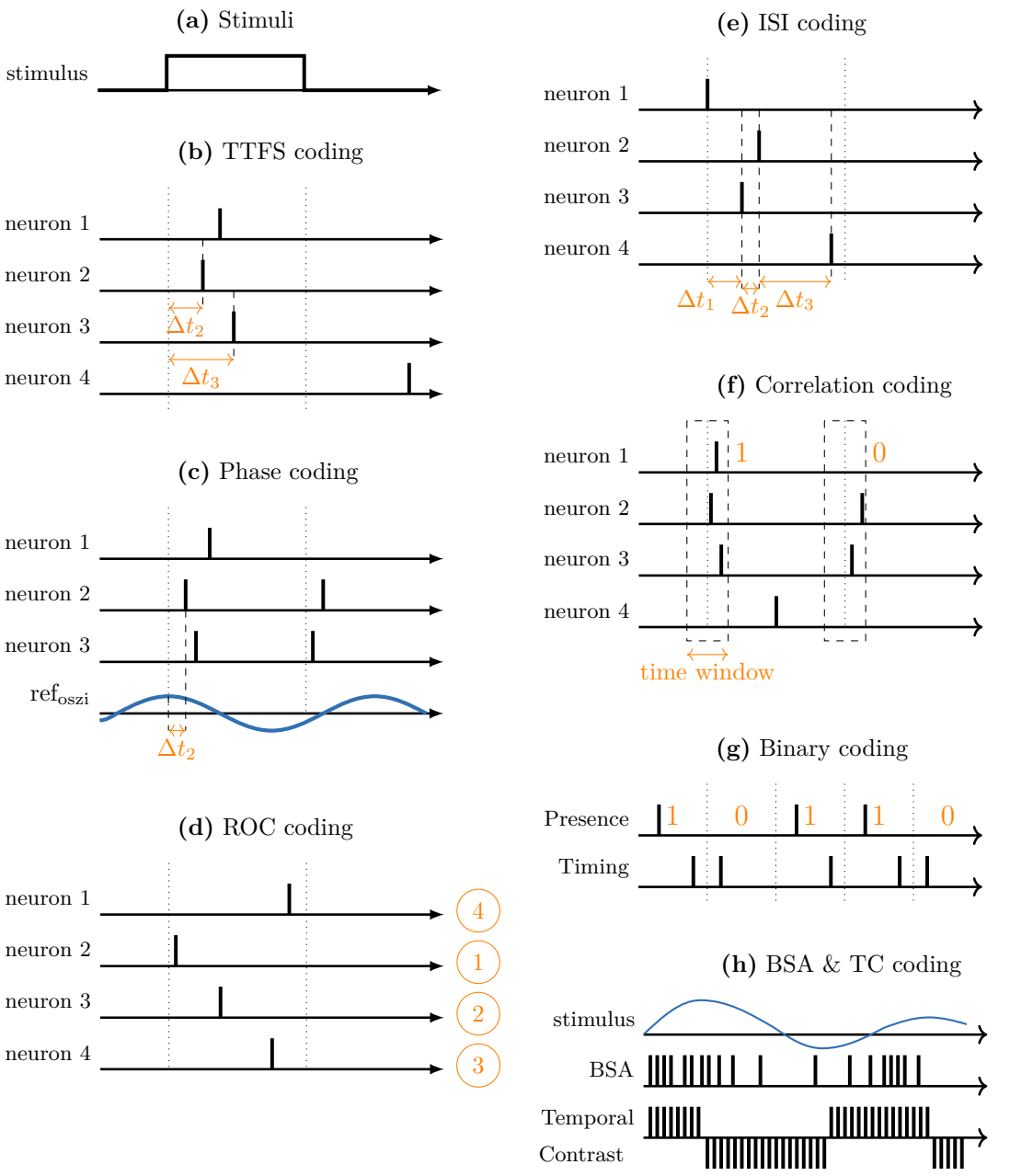

Figure 3.2: Temporal coding techniques: Overview of different temporal encoding techniques with the dashed vertical line as a reference. Here, Δt describes the temporal difference between reference and spike. Modified from our previously publication[39].

3.3.2 Temporal Contrast

The last subgroup we mention in this work is TC. The spike trains represent the temporal deviations of an analog signal [143], enabling the conversion of continuous analog signals. One example application is the event-based camera where TC encodes the light intensity changes with a high dynamic range [192], [193]. This biological-inspired camera only generates positive or negative events for each pixel during an intensity change that exceeds a predefined threshold. The following describes the theory of the three most famous TC encoding methods. Since all methods generate a positive and negative spike train $\delta[t]$, we define the spike function of the input signal $x[t]$ as:

$$\delta[t] = \begin{cases} +1, \text{ if } x[t] > +\vartheta, \\ -1, \text{ if } x[t] < -\vartheta, \\ 0, else, \end{cases} \tag{3.3}$$

where ϑ represents a static or dynamic defined threshold depending on the TC method, which we further explain in detail.

Threshold-Based Representation (TBR): uses a global threshold, which is defined by analyzing the signal mean and standard deviation (SD) with a scaling factor β. The threshold ϑ is defined:

$$\vartheta = \text{mean}(x'[0:t]) + \beta \cdot \text{SD}(x[0:t]), \tag{3.4}$$

for a selected time interval $x[0:t]$ of the signal's first derivative, which we can realize as hardware implementation by the deviation between $x[t]$ and $x[t-1]$. When the difference between the current $x[t]$ and the last sample $x[t-1]$ exceeds the threshold ϑ, then a positive spike appears as it was defined in Equation 3.3.

Moving Window: selects the threshold based on the historical mean of a moving window and a predefined accepted SD. Hence, we define the positive and negative thresholds like:

$$\pm\vartheta[t] = mean(x[t - t_w : t]) \pm \text{SD}, \tag{3.5}$$

where t_w is the length of a time window. The algorithms generate the corresponding spike when the signal exceeds the range defined by the threshold due to the positive and negative SD. A small SD increases the sensitivity and the spike count. Since the spike count correlates with the power consumption of event-based implementation, the SD directly influences the system's efficiency.

Step Forward: is another temporal contrast method that defines the threshold based on the previous value. The positive and negative threshold is defined by:

$$\vartheta[t] = \text{base} \pm \text{SD}, \tag{3.6}$$

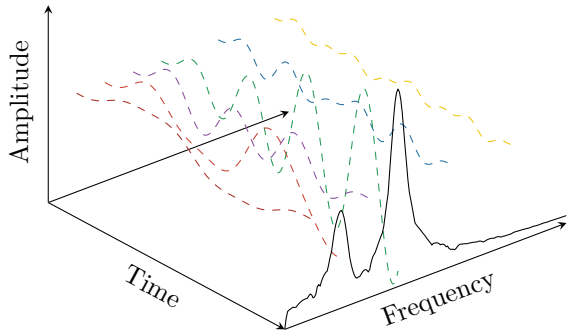

Figure 3.3: R&F frequency selectivity: Response of the subthreshold dynamics of the R&F neurons to a simulated superposition of two frequencies. Integration of the neuron states is similar to the Fourier transformation because we neglected the rest and spike function.

where the previously emitted spike sets the base. An action potential generates the update of the base by the positive or negative SD depending on the spike polarity. Compared to a pure comparison to previous values, it can better represent changes across multiple time steps. Hence, the standard deviation scales the algorithm sensitivity and the number of spikes. The SD greatly affects the overall encoding performance and removal of unnecessary information. Figure 3.2 demonstrates an example of TC encoding based on Step-Forward (SF).

Temporal contrast methods use the signal's temporal deviations by different threshold definitions and, therefore, affect the information in the spike domain. The challenge is to select suitable parameters for a good relation between information loss and the number of spikes. Important to mention are the high temporal and low spatial dimensions of this encoding scheme and that the individual factor depends on the application sampling frequency. For further information about TC encoding, we refer to [194].

3.4 Resonate-and-Fire Encoding

The following chapter exceeds the survey of encoding methods and proposes the R&F encoding, which takes advantage of the inherent neuron properties. We described the neuron dynamics in Chapter 2.2. A population of the R&F neurons acts like a frequency-selective filter bank with a spike generation mechanism. The initial idea of using the R&F neurons as encoder states back to [195]. The encoding provides two interesting properties: 1) it is frequency selective, 2) it generates spikes based on the amplitude, and 3) represents the phase in the exact spike timing by a slight modification. In the following, we will describe the signal encoding technique with R&F and use synthetic examples to demonstrate possible applications.

3.4.1 Properties of the Damped Resonator

Instead of a spike train, the R&F receives the continuous input signal directly to the real and imaginary neuron state. In the following, we use the neuron to encode only a real signal. Assuming the input signal is a superposition of sinusoidal, then the individual neuron would react to the existence of the resonate frequency, meaning the neurons where the resonant frequency matches with the input signal increase its amplitude significantly. Figure 3.3 demonstrates the response of the R&F without spike function to a two-tone signal. Since the input signal matches at least two neurons, the amplitude of the neuron oscillation increases. Due to physical reasons, the frequency is not ideal and spectral leakage appears. The temporal summation of the neuron states generates a Fourier transformation, as indicated in Figure 3.3, showing a resonant and non-resonant excitation.

A resonant excitation of a neuron n with resonant frequency ω_n exists if the input current is a sinusoidal oscillation, matching the resonant frequency of the neuron. Therefore, the input current $i[t] = a\ \cos(\omega_n t)$ enhances the neuron state. The main information is in the envelope of the transient response because, in a resonant case, it increases exponentially due to the damping component. Thus, we can define the envelope of the transient response

$$e_{0n}[t] = \frac{-A}{2b}\,(1 - e^{bt}), \tag{3.7}$$

with b as the damping constant with the condition $b < 0$. The amplitude A of the sinusoidal oscillation defines the maximum reachable amplitude of the neuron state. Thus, the amplitude of the neuron state is independent of the resonant frequency, but it depends heavily on the input signal amplitude. Equation 3.7 can be approximated linearly for a small damping constant. Thus, the neuron's amplitude increases over time, which means the time is proportional to the signal's amplitude. The damping constant allows the encoding of continuous signals with frequency changes. While a frequency, when present, will result in the neuron's excitation, the neuron will lose excitation by the disappearance of the frequency because of the damping property. The damping leads to an exponential decay at the end of the resonant excitation with the envelope of:

$$e_{1n}[t] = A_{max}\ e^{bt}, \tag{3.8}$$

which depends on A_{max}, which is the maximum amplitude of the transient response. The maximum amplitude is defined by the $max(e_{0n})$ of Equation 3.7.

During a non-resonant excitation ($w \neq w_n$) of neuron n the envelope increases much lower in the same time. Since in any real signal wideband noise exists, the R&F neurons are always stimulated, differing in the amplitude of the excitation of the neuron. Also, effects like spectral leakage lead to increased excitation of neurons surrounding the frequency of the input signal, and the same resonating case is less probable.

(a) Original spike mechanism of Izhikevich

(b) Phase-dependent spike mechanism

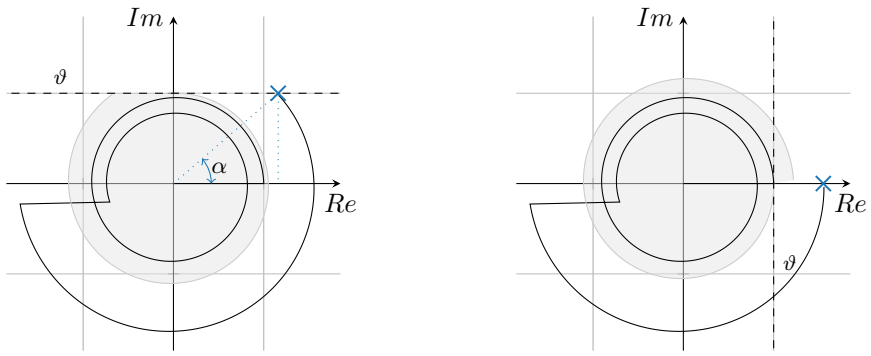

Figure 3.4: Comparison of spike methods: Comparison of the phase diagrams for different spike methods of the R&F neuron. The discrete model receives two input spike events to reach the threshold. Simulation is stopped at the spike generation, and no reset is applied here. The blue cross represents an action potential.

3.4.2 Properties of Spike Mechanism

One major component of a neuron is the spike mechanism, as already briefly described from an action potential point of view in Chapter 2.2. The spike carries the information in the rate or timing. In population-based encoding, the information is either in the rate of the population activity, the relative timing, or the combination of both. The R&F neuron transforms a temporal signal into a spatio-temporal representation by extracting the frequency components and simultaneously removing noise. It consists of two internal states, providing various variants of the spike and reset mechanism. In the following, we will analyze the different spike methods to show the suitability of R&F neurons for data encoding. We use the phase diagrams in Figure 3.4 to show the difference between the original and the phase-dependent spike mechanism.

The original spike mechanism proposed by Izhikevich [115] is defined by

$$\delta[t] = \begin{cases} 1 & \text{if } Im(\underline{z}[t]) > \vartheta \\ 0 & \text{else,} \end{cases} \tag{3.9}$$

that means if the imaginary component exceeds the static or adaptive threshold ϑ, an output spike is emitted, and the neuron is reset. The measure of the real component as a threshold indication is possible but not recommended because it is directly connected to the input current and can be affected by noise. Figure 3.4a shows the response to a spike stimulus with the static threshold defined as ϑ. For visibility reasons, we use the spike, but the same idea applies to a random input signal. If the neuron state exceeds the gray area, it will always generate a spike. [195] experienced the dependency of the phase and the amplitude during the simulations. The reason can be explained with the

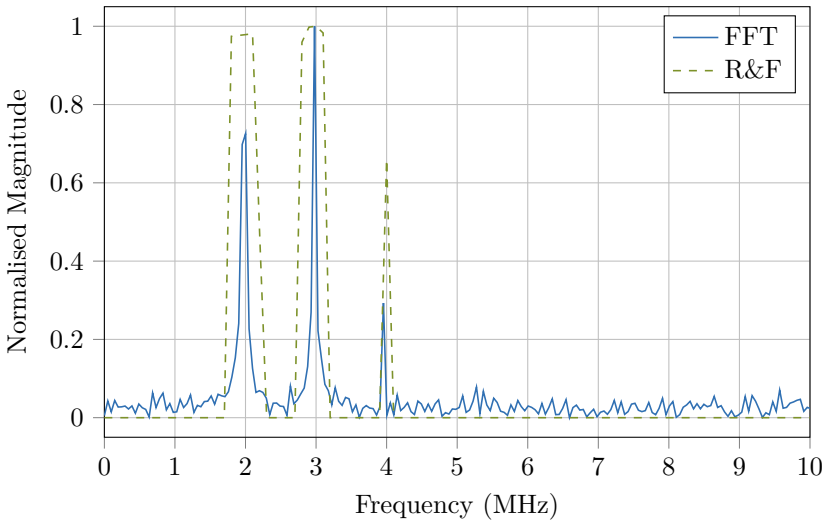

Figure 3.5: TTFS with R&F: Comparison of the FFT with the decoding of the first spike within a R&F population. We normalize the FFT and the temporal distance between the start of the simulation and the first spike of the individual neurons. An early spike event means a high amplitude, while no spike indicates no signal component. The sparse response uses a simulation with 512 time steps and the R&F neurons without reset.

phase diagram because the phase is defined by

$$\alpha = \arctan(\frac{\vartheta}{r}), \tag{3.10}$$

where r is the instantaneous amplitude of the oscillation. Assuming that we are interested in the phase at a fixed threshold potential, we observe that an amplitude change directly affects the phase, making it impossible to estimate the phase exactly. However, the time of the first spike is directly proportional to the amplitude and phase of the system, but the individual information cannot be extracted. Two signals with the same amplitude and frequency but with a phase difference of π would lead to a different spike timing where it is impossible to differentiate the phase or the amplitude. With the original spike mechanism of Izhikevich, it is impossible to detect the exact amplitude and phase of a signal in noisy environments [195].

Not only the first spike carries information about the amplitude but also the distance between spikes with a reset mechanism or the number of spikes. The correlation between amplitude and phase is only valid for the first spike but does not affect the time between spikes in a neuron with a reset mechanism. Since the generated spike does not always appear at the same position of the neuron's oscillation, an additional spike timing error appears that influences the amplitude estimation. Also, the number of spikes limits the detection of short signal components; even adaptive thresholds cannot overcome this issue, but they have the benefit of better distinguishing between low and high amplitude signals [43] with the drawback of additional computations. Another problem

of this threshold with binary spikes is the effect that close R&F neurons can lead to the same number of spikes because of spectral leakage. Thus, a signal with a constant frequency influences the surrounding frequencies through the leakage. Figure 3.5 depicts an example that compares the Fourier transformation with the R&F encoding and the TTFS decoding. Important to note is the noise suppression of the R&F by no activity of the not existing frequencies. Also, the figure indicates at 2 Hz the problem of the spikes from the surrounding neurons. Nevertheless, we demonstrate with this simulation the extraction of the amplitude information by the time of the first spike.

One way to retrieve more information are weighted spikes with $Re(\underline{z[t]})$ as amplitude [118]. We can use the Pythagorean equation to extract the amplitude $r = \sqrt{Re(\underline{z[t]})^2 + \vartheta^2}$ and the basic trigonometric conversions to define the angle $\alpha = \arcsin(\frac{\vartheta}{Re(\underline{z[t]})})$. However, this encoding does not use the exact spike timing and increases the necessary complexity to extract the amplitude and phase.

Another approach is the utilization of the exact timing of a spike. Thus, we define a phase-dependent spike mechanism:

$$\delta[t] = \begin{cases} 1 & \text{if } Im(\underline{z}[t]) > \vartheta \ \&\ Re(\underline{z}[t]) = 0, \\ 0 & \text{else,} \end{cases} \tag{3.11}$$

it means that the spike is always at the zero transition of the imaginary axis, where we have a 90° phase difference between real and imaginary. At this point, the imaginary component is maximized. The spike timing carries the phase information, and the ISI to a reference spike contains the relative phase. Important to note is the relation of the phase, frequency, and time between spikes. Figure 3.4b shows the corresponding phase diagram with the spike at the zero transition. The amplitude or magnitude of the signal cannot be exactly defined through the number of generated spikes but with weighted spikes:

$$\delta_{weighted}[t] = (Im(\underline{z}[t]) - \vartheta)\, \delta[t]. \tag{3.12}$$

This exact spiking of the neuron has the benefit that we can estimate the frequency, amplitude, and phase of the signal. The frequency is encoded twice in the spatial domain of the spike and the time between two consecutive spikes. The amplitude is in the weight of the signal, and the phase is in the timing of the spike to a global reference. Recently, [118] showed the encoding with R&F neurons with a similar threshold idea, but it uses the condition of the transition of the real axis and the threshold on the real component. It means that the spike always indicates the zero phase, but we tried to avoid the direct connection of the threshold with the input stimuli to reduce noise-related effects. The threshold defines the sensitivity to low amplitude signals such as noise, and the spikes focus on the higher amplitude signals. Compared to the Fourier transformation, it removes the noise components and enables sparse communication. [118] compared the phase-dependent spike mechanism with the Short-Time Fourier Transform (STFT) and observed the lower communication cost on the Loihi 2 neuromorphic hardware. There are even other variants to define the threshold. [119] implements a threshold phasor associative memory with

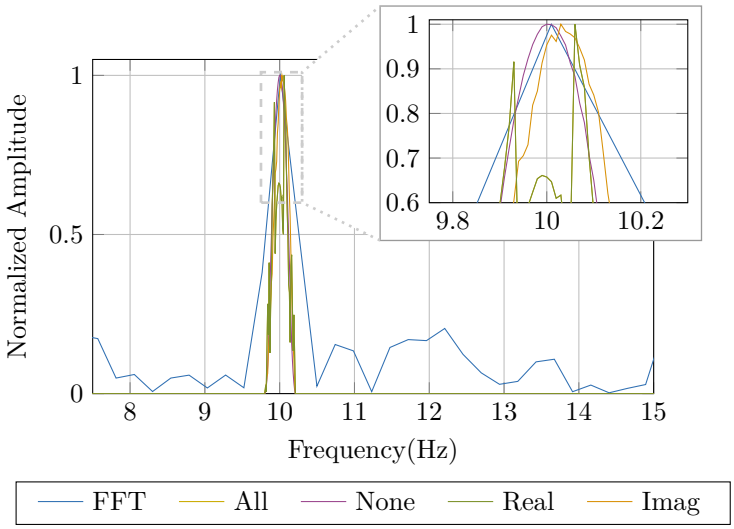

Figure 3.6: R&F frequency-spectrum: Time-to-Frequency conversion with FFT and R&F neurons with different reset Methods. The reset method all and real overlay each other. The right plot focuses on the region of the 10 Hz signal.

R&F neurons that use two threshold conditions: $Im(\underline{z}[t]) > \vartheta \,\&\, Re(\underline{z}[t]) > 0.2$ for spike based data.

3.4.3 Properties of Reset Mechanism

An additional feature of all biological neurons is the rest functionality after an action potential or spike. Neuroscientists observed that the potential falls below the resting to provide an additional time that reduces the likelihood of a new spike generation called refractory [21]. However, this mechanism is less relevant in R&F neurons since the neurons have to be stimulated again to oscillate.

The R&F neuron has two internal states to reset the neuron, the imaginary and real components. Therefore, we have at least three different possibilities to reset the neuron by:

$$\underline{z}[t] = \begin{cases} j\, Im(\underline{z}[t]) \text{ or,} \\ Re(\underline{z}[t]) \text{ or,} \\ 0 \end{cases} \tag{3.13}$$

where even the reset value can be adjusted, for example, to the threshold potential ϑ. Due to the resonating effect, a negative value does not prevent the firing of the neuron and is, therefore, not further analyzed. After resetting a single component, the neuron states continuous to resonate, which is an interesting property for spike processing. In the application of encoders, it leads to an unbalanced spike pattern around the center frequency compared to the complete reset for non-binary spike events. Also, the complete reset reduces the amount of spikes in the processing chain.

Figure 3.6 shows the accumulated spikes of different reset methods. We use the previously explained weighted phase-dependent spike mechanism and the discrete neuron implementation. The synthetic input consists of a sinusoidal oscillation with 10 Hz and zero phases with a Signal-to-Noise Ratio (SNR) of 1 dB. We use here additive white Gaussian noise to model a balanced distribution. The accumulation of the weighted action potential carries the information about the existing frequency components in the signal. However, the overall results depend on the neuron threshold and the reset mechanism. The threshold leads to direct noise and low amplitude suppression. The real peak can be detected best with no reset mechanism (none), with the highest spike count of 909. The lowest spike count of 102 has the worst performance, as seen by the reset of the real (real) or the real and imaginary (all) state, because it is affected by the input noise, as we could observe for multiple simulations. The reset of the imaginary state provides 733 spikes and a decent performance. As expected, the spike count is also affected by the number of neurons and the spacing between the resonance frequencies. Here the simulation uses 2000 neurons from 0 to 20 Hz with a frequency step size of 0.01 Hz, where only 2% of the neurons generate spikes. The selection of an appropriate mechanism is not a simple decision, but these findings already give insides into the variants, which we further evaluate in the applications.

3.4.4 Comparison to the Fourier Transformation

Previously, we have described the capability of the R&F neurons to convert an input signal into a frequency-time representation by utilizing the inherent property of the resonant frequency kernel, as depicted in Figure 3.3. Also, we compared the response with the Fourier transformation. The following section will present an extended comparison between R&F encoding and STFT methods by investigating the computational complexity and the output bandwidth. For a detailed description of the STFT, we refer to [196].

Equation 2.23 defines the discrete form of the R&F. In case of no reset mechanism and the initial zero condition of the internal states $z_k[0] = 0$, we can reformulate the equation into the sum of the individual multiplications of the input with the oscillation kernel:

$$z_k[n] = \sum_{n=0}^{N} \mathrm{e}^{\Delta t\, b_k}\, \mathrm{e}^{\mathrm{j}\,\Delta t\, \omega_k}\, a_k[n]. \tag{3.14}$$

This form resembles the STFT described in [118]. The damping term $\mathrm{e}^{\Delta t\, b_k}$ defines a low pass filter naturally as an exponential decay, which corresponds to the windowing function of the STFT. The resonant frequency ω_k defines the rotation factor by the complex exponential. Hence, a population of R&F neurons can perform a similar function as the STFT.

Figure 3.7 indicates the correlation of the spike response of the R&F neurons to the time-domain input. Here, we transform the instantaneous frequency of $f(x)$ to the time-domain signal and apply the R&F encoding. The spike response has a similar shape as the expected frequency course, but the spike count depends on the steepness of the frequency change. We used a low-frequency input signal and 12 R&F neurons with a

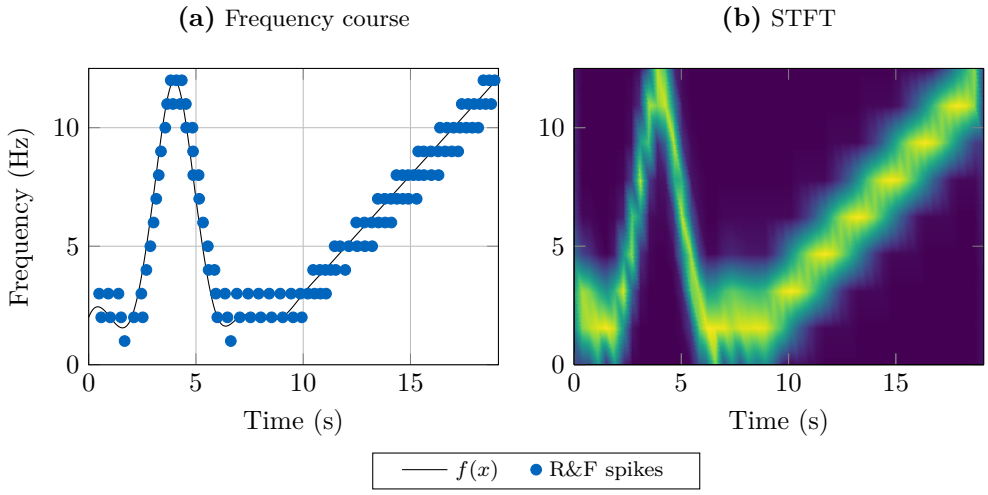

Figure 3.7: Instantaneous frequency of R&F and STFT: (a) indicates the generated frequency course of the input signal and the corresponding spike response of the R&F neurons. The neurons use the phase-dependent spike mechanism with a damping constant of -1 and a threshold of 0.1. In total, 113 action potentials carry the $3,800$ time-step long signal information. (b) illustrates the STFT with zeroing and a FFT length of 128 and an overlap of 50 step.

linear scaled resonant frequency in this example. The damping factor is -1, and the threshold is 0.1. A smaller resonant frequency scale and a higher threshold could achieve a better resolution.

In the case of the R&F encoding, the temporal resolution correlates with the simulation time steps because it uses the inherent decaying as a window function. Another important aspect is the sparse output events of the R&F neurons compared to the STFT [118]. In Figure 3.7, a total of 113 spikes across 12 neurons out of 20 carry the information of the input series. The STFT uses a window and FFT length of 128, but we reduce it to the frequencies of interest. Therefore, in the best case without overlapping windows, the complete output consists of 403 complex values for 13 frequency bins. In the worst case, the spectrogram is defined by every time step and generates $49,413$ complex values. Thus, the sparse representation reduces the number of output values by a factor of $875x$ in the given example because the phase information is encoded at the time of the spikes instead of an additional floating point value.

We can pre-calculate the network parameters as shown in Equation 2.24. Therefore, the network consists of a total of 25 parameters, and each neuron has four multiplications, three additions, the step-activation function, and the reset function. The activation function consists of a single branch instruction, and the reset replaces the internal states with zero if necessary. The complexity of the R&F algorithm is $O(MN)$, where M is the number of desired spectral components and N corresponds to the number of time steps.

A baseline method to extract the frequency course is the STFT, which applies the Fourier transformation on a moving window across the time series [197]. It can either be

Table 3.1: Computational complexity of STFT algorithms. N is the number of input time steps, and M describes the number of desired spectral components. L refers to the length of the windowed Fourier transformation, and the highest number of windows equals N for a moving window with a single step and zero padding.

STFT	**Complexity**	**Multiplications**	**Additions**
Algorithm		*Per spectral component*	
DFT*	$O(MN)$	$2N$	$2(N-1)$
FFT*	$O(N \log_2 N)$	$2 \log_2 N$	$3 \log_2 N$
Goertzel*	$O(MN)$	$(N+2)$	$(2N+2)$
DFT**	$O(MLN)$	$2LN$	$2N(L-1)$
FFT**	$O(NL \log_2 L)$	$2N \log_2 L$	$3N \log_2 L$
Goertzel**	$O(MLN)$	$N(L+2)$	$N(2L+2)$
R&F	O(MN)	4N	3N

*without window overlap.

**with single-step moving window.

implemented as DFT, FFT, or by the Goertzel algorithm [198] with different complexities as shown in Table 3.1, where we demonstrate the trade-off between the temporal resolution and the number of operations. The best case consists of non-overlapping moving windows with a length L, and the worst case appears for a stride of one with zero padding because the number of windows equals the length N of the input data. M refers to the number of spectral components.

3.5 Discussion and Summary

Over thousands of years, nature has evolved information encoding of sensory information for efficient processing in the brain. Thus, the encoding has the following requirements: 1) information reduction, 2) noise suppression, 3) representation in rate or temporal spike events, 4) energy efficiency, and 5) real-time conversion. To solve all of these requirements, nature has developed different codes.

This chapter analyzed current state-of-the-art encoding methods using rate or temporal codes. The rate codes use the firing frequency as the primary information source, and such networks can be converted from traditional ANNs [23]. However, the rate codes suffer in their energy efficiency because each spike is processed so that the number of spikes correlates with the power consumption in event-based systems. One solution to the energy problem is the use of temporal spikes where the information is at the time of the spike event.

The real-time conversion requirement increases the complexity of the encoding mechanism. Methods like count rate or TTFS need time to represent the information. Thus, the information is represented in subsequences which is not applicable to high-frequency signals like in radar. Methods like TC are more promising because they can continuously

generate spikes depending on the input changes. Also, a population where each neuron is responsive to a different amplitude level would be sufficient, but not every population encoding is suitable. The population code is vaguely defined since every code can be extended to a population code. A receptive field like Gaussian kernel filters the information, represented as TTFS or rates [199], [200].

We extended the original review [39] with the R&F encoding that combines a frequency-selective preprocessing with the conversion to spikes. Thus, we classify the R&F encoding as a temporal filter-based method. The resonating properties allow the conversion of the single-dimensional input to a higher dimensionality while suppressing noise. The original signal's information is compressed in the spikes appearances, timings, and rates. Thus, the encoding combines the idea of population, temporal, and rate codes, providing a high information density. This idea is inspired by observations from the auditorial cochlear [201]. The R&F encoding is comparable to the STFT processing with a sparse stream processing. However, the current neuromorphic hardware implementations do not fully support the R&F except for the Loihi 2 and SpiNNaker 2.

This survey about different encoding methods contributes to the main research question and provides a theoretical answer to RQ 1 and 2. We reuse the R&F encoding in all applications.

4 Radar Interference Detection

One application to demonstrate the differences between the encoding schemes is the pattern detection of temporal data or time-series classification. Temporal data underlie changes over time based on the historical temporal context [202]. We use the discrete representation of a time series to model the system, which could be applied in continuous applications. Time-series analysis is essential in various applications like finance, economics, server resource management, and much more [203], [204]. Often, the analysis is not real-time crucial and can be calculated offline, which relaxes the memory and processing requirements. In the case of radar-to-radar interference, the detection should be real-time capable with a low power budget because a vehicle could have a traveling speed of 130 km/h or even more and limited battery power. Therefore, we investigate SNN architectures with different encoding schemes to detect interference by analyzing the time-domain signal after the ADC.

Further information about radar interference is provided in Chapter 2.1.3, and the background of encoding is covered in Chapter 3. Also, we compare the proposed architectures with the traditional threshold technique, which utilizes the assumption that interference always has a higher amplitude than the signal components. We demonstrate two basic supervised Machine Learning (ML) methods for classifying interference: 1) deviation detection and 2) pattern detection. The detection of interference is a sort of outlier detection with knowledge of the interference properties, which makes it suitable for supervised learning of application-specific outliers [202]. In safety-related applications, a supervised approach provides more control than unsupervised methods. Additionally, we optimize the network architecture to process spatio-temporal patterns efficiently. Therefore, we generate a synthetic and semi-synthetic dataset and compare our proposed method with traditional and deep learning-inspired techniques like the CNN and Long Short-Term Memory (LSTM).

4.1 Related Work

In the following, we will provide an overview of existing methods to detect interference that we also use as a baseline. Some detection algorithms are closely interwoven with the mitigation, but other mitigation methods are investigated based on the ideal detection approach. The detection is the first step before applying any mitigation method, signal processing adaptation, or forwarding the information of no-functionality to the driver. Consequently, we provide an overview of classical deterministic and traditional machine learning methods and highlight the limitations of these approaches.

It is widespread to use CFAR as an algorithm to detect the peaks within the range-FFT or the RDM [205]. The algorithm analyzes the input data by a moving mean across a

predefined window shape. It, therefore, differentiates noise from a signal component, but it has the drawback that if a target is below the noise due to interference, then the target will not be detected. An extension of this idea is using the time-frequency information where CFAR is applied across each frequency, and the generated map is used to mask interference. This approach takes advantage of the changing frequency of the interference compared to the static target with constant frequency throughout a single chirp [206].

The most basic interference detection method is a simple threshold that detects if the received signal crosses the expected working range; this technique is also called clipping [207]. It relies on the assumption that the amplitude of interference is always higher than the signal reflections. Consequently, this threshold type detects only saturation for very strong interference or a close, highly reflective target that does not fit the sensor's configuration. We name this method thr-simple in the following chapters. Also possible is the usage of the standard deviation across a sliding window. The outlier score of a wideband interference is more likely to have a higher variation because of the high-frequency components than the target echoes, and interference leads to a higher amplitude than the expected targets. A predefined threshold based on the radar application and settings differentiates interference from the expected signals. We refer to this method as thr-windowed. Another idea of the threshold method is to estimate the threshold based on the strongest target reflections iteratively. The threshold is iteratively defined as the weighted mean of the signal without disturbances that are neglected due to the previous threshold. This process is repeated until the threshold change is below a predefined value. We implemented the newest version of this iterative threshold [208] and called it thr-iterative. An additional threshold-based method calculates the amplitude variations between two consecutive samples and compares them with a predefined threshold [209]. The threshold is selected based on the radar application, the expected target reflections, and depending on the sampling frequency. The patent already indicates the problem of wrong detections due to the nature of the static threshold if the leakage between the transmit and the receive antenna is too high or the interference amplitude is not higher than expected by the algorithm.

Another way to detect interference operates with the negative and positive frequency characteristics of out-phase interference, which only applies to complex basebands. Therefore, every channel consists of a real and imaginary component by applying a 90° phase rotation. Because a real target reflection only appears in the positive half-frequency spectrum, the interference is detectable in the negative frequency component. There are different techniques to use the detections by interpolating the samples during the duration of the interference and applying to zero or an adaptive noise canceler [210]. This method increases the product cost heavily due to the higher silicon area consumption.

The recent trend of NN is also visible in the topic of radar interference detection and mitigation, where various methods are applied to different steps of the signal processing pipeline. [211] applies a network of Gated Recurrent Unit (GRU) cells to detect and mitigate interference in the time-domain. The recurrently connected network acts as a filter to remove interference by training the network to reconstruct the time-domain data without interference. This approach is fully supervised and, therefore, good to control. However, it is difficult to understand what the networks learn compared to autoencoder

architectures, where the latent space represents the extracted features from the input. [212] demonstrates the use of an image processing-inspired CNN autoencoder for RDM denoising. The denoising can remove interference and general noise from the RDM. The architecture is fixed to the settings of the radar sensor, and a change in the number of ramps affects the size of the RDM. Hence, the network has to be retrained for each individual setup.

During training, the network optimizes the image-to-image translation of corrupted RDMs. Therefore, corrupted and not corrupted samples are necessary. The authors recorded the interference IF signal during the off phase of the transmit path of the victim sensor and performed a linear superposition of the recorded data. There are two ways how the CNN can denoise the maps, either by acting as a sort of CFAR detector or by investigating the changes across the Doppler dimensions because the interference is unlikely to be the same across multiple chirps, but this was not further investigated. Another CNN-based network classifies the interference into different classes depending on the source of the disturber [213]. Similarly, [214] classifies the type of interference with a support vector machine on the RDM. However, the source of the interference is less important than the actual detection of the corrupted chirp.

4.2 Setups

The following chapter focuses on the two methods used as interference detectors by assuming interference as an outlier and classifying the interference pattern. We explain the central concept and the training methods used for the different network architectures. This work points out the properties of the individual encoding techniques based on the two central concepts. Due to the inherent differences between the two methods, we have to use different datasets, limiting the direct performance comparison.

4.2.1 Normality Estimation

The first method assumes that interference is an outlier from the normal receive signal of the radar sensors. Therefore, we use a predictor to represent the expected normality and compare it with the real value. In the following, we will use the word outlier as a synonym for interference. Such a method is widespread in situations with sparse outlier events and unknown properties of the outliers [215]. This concept was presented in the publication [40], where we showed an LSTM-based predictor in combination with a zeroing mitigation method that masks the detected outlier sample points.

The overall idea is shown in Figure 4.1 with the continuous input time series $x[t]$ and the corresponding output $\hat{y}[t]$ that indicates the existence of an outlier. The predictor estimates the next time step $\hat{x}[t+1]$ based on the historical input values and calculates the outlier score with the delayed predictor output:

$$s[t] = x[t] - \hat{x}[t]. \tag{4.1}$$

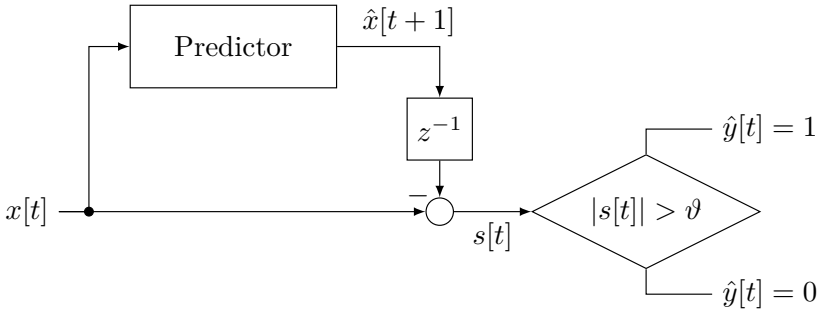

Figure 4.1: Concept of semi-supervised outlier detector: The predictor estimates the future sample points by using a local temporal memory to extract historical relationships. The outliers are detected by the amount of deviation between the expected and the real data. 1 indicates a detected outlier, and 0 no outlier.

Due to the simplicity and similar performance, we decided to use a static threshold ϑ for differentiation between normality and outlier:

$$outlier(t) = \begin{cases} 1, & \text{if } |score(t)| > \vartheta \\ 0, & \text{otherwise} \end{cases} . \tag{4.2}$$

The threshold correlates with the system accuracy because a lower threshold increases the probability of false-positive detection, and a higher threshold reduces true-positive detections. A good starting point for the threshold value is above the maximum training error between the predicted and real output, but a hyperparameter search provides an even better selection. Dynamic threshold methods with a higher computational cost can achieve a higher complexity and slightly better performance.

Training: The training method of the one-step-ahead predictor differs from traditional classification and regression problems because we take advantage of the temporal properties of the input data. The network receives and generates one data sample per time step. Therefore, we use BPTT with an adaptation for sequence-to-scalar supervised prediction, but we test the whole model in an unsupervised mode. This approach is called semi-supervised because the predictor is trained and supervised, but the complete system is used unsupervised. This technique has the advantage of a high training control but benefits for unsupervised properties during inference by detecting any outliers from the expected normality. We depicted the training and test data pre-processing in Figure 4.2, where the training sequences are separated into subsequences by applying a moving window. In our case, we use the last value of the chunks as the target value y, for a one-step-ahead prediction. The window length defines the number of unrolled layers in time and is a trade-off between vanishing gradient for long windows and the capability to learn temporal features. Hence, we use a window length of 100 time-steps and a single

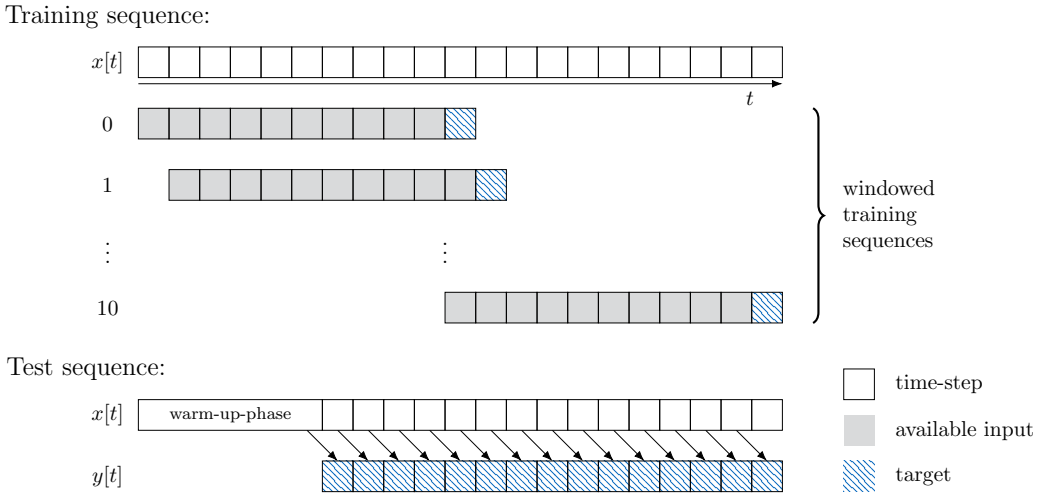

Figure 4.2: Training sequence-to-scalar: From the training sequences, subsequences are generated using a moving window as depicted by the windowed training sequences. The available input is the step-wise input to the network, and the output is the next predicted value. The test sequences use a warm-up phase to stabilize the network and then continuously predicts the target value $y[t]$.

target value in our example. The sequence-to-sequence method also exists, first developed for natural language processing [216]. The idea is to predict the target sequences instead of a single scalar value. However, the network will not be forced to learn temporal dependencies between the first and the last sample point, which is crucial in the given application. The same approach can be used for time series analysis with a different input and target sequence, like observing the current consumption of a system and predicting its state.

We do not apply any windowing for the test and validation, as shown in Figure 4.2. After the warm-up phase, the network has stabilized, and the output can be used to predict the target value. In our case, the test sequences contain an additional chirp which we remove before the evaluation. We use the Mean-Squared-Error (MSE) loss between each prediction $\hat{y}_i$ and target y_i, which is defined by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{4.3}$$

with n number of predictions. The loss function correlates with the outlier score and shows that a high MSE leads to a low outlier detector performance.

Architectures: The baseline architecture consists of LSTMs neurons, which have inherent recurrency by their gated architecture with an input, output, and forget gate [217], [218]. These gates reduce the vanishing gradient problem by individual activation

functions but increase the computational cost heavily, as shown by the equations [219]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4.4}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4.5}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4.6}$$
$$c'_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4.7}$$
$$c_t = f_t c_{t-1} + i_t c'_t \tag{4.8}$$
$$h_t = o_t \tanh(c_t), \tag{4.9}$$

where σ is the sigmoid activation function and i, f, and o are the input, forget, and output gates, respectively. The input x and the hidden state h are multiplied by the weights W and U. We use two fully connected hidden LSTM layers with 25 recurrent connected cells for the one-step-ahead prediction task. The input layer is a single input node of the corresponding radar signal for a single receive channel. The output combines the LSTM output by a dense layer with one output neuron without activation function. An activation function at the output negatively influences the prediction by binding the output into the value range of the function. An example is ReLU, where negative output is bound to zero; therefore, a negative prediction is impossible. Additionally, we replaced the LSTM cells with GRU cells for improved computational complexity.

We use the step-forward temporal contrast method for the spiking network to encode spikes and decode the spike events. We motivate the use of TC by the continuous change and periodicity of the radar signal. Hence, each input sample point corresponds to a time step of the network. Therefore, a pure rate-based encoding cannot react to fast changes, and a temporal coding like TTFS would need sub-time steps to represent the signal's amplitude better. Filter-based methods like Ben's spiker algorithm (BSA) or Hough spiker algorithm (HSA) have proven their existence in different applications like ECG classification [220]. However, [194] has shown that temporal contrast performs better on the step and smooth inputs, and the subcategories of SF perform best for the given task.

Additionally, we directly inject the data as input current to a population of LIF neurons and adjust the weights during training to create different neuron transfer functions. Further details about the encoding methods are in Chapter 3. Therefore, we compare the baseline architecture with the SNN architecture with SF, or current injection encoding, a population of LIF neurons, and a trainable temporal contrast decoder based on a linear superposition of the network output. We use the same two-layer architecture with 25 hidden neurons for both encoding schemes.

In this binary classification task, we use the accuracy metric normally defined by the division of true positive detection by total possible detections. In unbalanced datasets, this can lead to misinterpretation of the results; therefore, we apply the balanced accuracy defined by [221], which considers the class distribution in a binary classification task. The balanced accuracy is specified by

$$\text{accuracy}_{\text{balanced}} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right), \tag{4.10}$$

where TP, FN, TN, and FP are the true positive, false negative, true negative, and false positive detections. Further information about these standard stochastic metrics can be found in [222]. The dataset consists of 51.4% samples with interference, but since we compare here the sample points and not the single time series, the class distribution is highly unbalanced. Only 1.5% of the sample points in the complete dataset are labeled as outliers. However, the network's performance is also falsely reduced because not all interference sample points are detected within the expected window. The detected outlier samples are smoothed by convolving the outliers with an impulse function. The duration of the impulse function is predefined to 10 and is either zero or one. Afterward, the results are clipped to represent just the two classes of normal and outlier. With this, we increase the number of detected samples and the likelihood of true positive detections, which is especially important for the additional zeroing mitigation. Mitigation is a technique to remove the effect of interference either by interpolation or removing the affected samples. Here, we use exemplary zeroing mitigation by replacing the corrupted samples with zero. In the method evaluation, we will analyze the prediction performance separately, the outlier classification of the samples, and on a chirp basis to provide an understanding of how well the proposed method performs.

4.2.2 Pattern Classification

The second method assumes that interference follows a specific pattern, as described in Chapter 2.1. Thus, we can apply a pattern classification method to differentiate between normality and interference. In this work, we have limited the detection to binary classification due to the limited availability of sample data. However, this concept is extendable to more outlier patterns and gives first insights into a possible signal processing with SNNs. The architectural changes provide a study about the suitability of the different encoding techniques for processing radar data.

All the architectures are trained with BPTT and the ADAM optimizer [17]. We apply early stopping, which means the training is halted when the validation loss has not improved for a specified duration (called patience). All architectures are mainly trained on the semi-synthetic interference dataset where each chirp is labeled as normal or interference, otherwise explicitly mentioned. We use a logarithmic Softmax function during the supervised training for the Negative Log Likelihood (NLL) loss function. However, for the visualization, we use a normal Softmax function. In all SNN architectures, we utilize the surrogate gradient because of its high robustness [38]. The simulations are done with the Norse simulator [223] and a custom-designed R&F and TC implementation in PyTorch [117].

Figure 4.3 provides an overview of the different architectures evaluated in the following chapter. The digital input signal is the ADC samples of the time-domain radar signal we forward to every method under test.

One baseline architecture is inspired by the state-of-the-art by using a CNN to classify interference instead of denoising [224]. Therefore, the input stream is converted into a vector representation by buffering the sample points. A one-dimensional convolution layer with multiple channels and ReLU activation extracts the features from the time series.

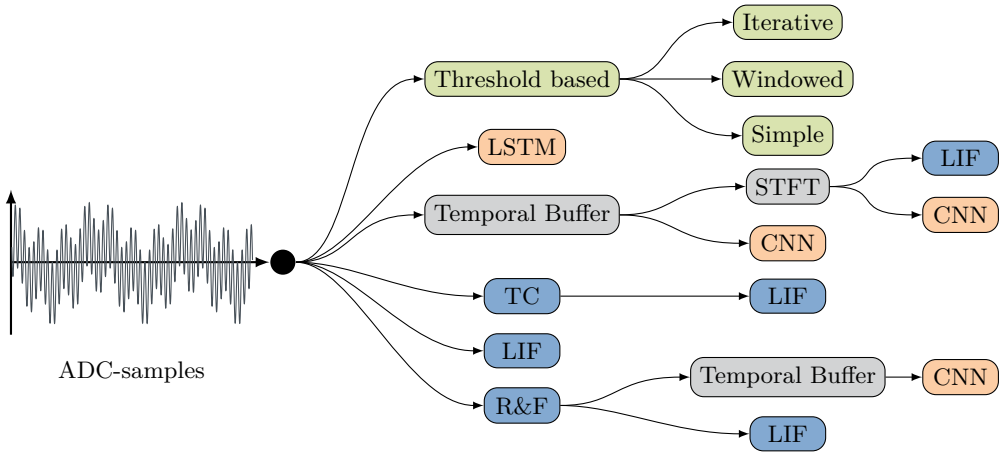

Figure 4.3: Classification architectures: Overview of all basic architectures for evaluating the interference classification task. The input is the temporal ADC-samples of the radar signal. Green indicates the state-of-the-art methods, orange is the ANN, blue is the SNN, and gray is the general components. The temporal buffer converts the temporal sequence into a frame-based representation.

The convolutional layer acts like an averaging filter with a kernel size of 10, a stride of 8, and 4 output channels. Figure 4.3 depicts the architecture without considering the exact shapes and sizes reported in Appendix A. Afterward, the channels' output is flattened and fed into a fully connected dense layer consisting of 28 neurons with ReLU activation. The readout layer with fully connected synapses uses two output neurons. Inspired by the normality estimation technique, we apply a network of 28 LSTM cells directly on the radar signal. Therefore, the network consists of an input and readout layer, where the input should extract the important features across the temporal dimension, and the readout classifies the activity.

Here, we extend the previously published SNN classifier, which uses the frequency selective R&F encoding [41]. So, the resonate-frequencies are linearly spaced across the supported IF of the radar sensor because the frequency corresponds to the range information. Thus, the frequencies are within the interval from $2\,MHz$ to $24\,MHz$ with a step size of $0.105\,MHz$ and a damping factor of -2. We use two different spike generation methods, the binary, and graded version, to analyze the additional information content of the graded events. After an action potential, the neuron is reset by setting the real component to zero. The encoding layer is fully connected with the hidden population of recurrent connected LIF neurons to extract the spatial and temporal correlation of the spike events. The reset to zero of the membrane potential is applied after each spike event of the individual LIF neuron. The readout layer consists of two neurons for each class and uses LI neurons. Thus, the output neuron integrates incoming events, and a higher membrane potential indicates the existence of the corresponding class. This task focuses on the differentiation between a normal or disturbed chirp and does not detect the exact position of the interference. Therefore, we use the last time step of the readout layer for the classification, which relaxes the requirements of the time constant, and a

non-leaky integrator could also be used. The initial time constants of the hidden layer are adjusted to the time step size of the given dataset.

Besides, we apply the famous current injection and the TC step-forward encoding as a replacement for the R&F encoding, which also introduces fundamental architectural changes because of the reduced spatial dimension after the encoding. The STFT has similarities with the R&F encoding and is another exciting way to pre-process the data. The CNN can directly use the two-dimensional output of the R&F and STFT, but the SNN needs an additional encoding. Therefore, we reuse our previous investigation of ways to encode RDMs into spikes using a binarization based on the data points above the mean [44].

4.3 Datasets

For the evaluation of the interference detection methods, we generate different datasets that will be described in the following. Since interference is sparse and difficult to catch in real-world applications with a few radar sensors, we mainly use synthetic datasets. Synthetic datasets [225] are a crucial element of NN optimization and have multiple advantages, like the number of samples, dataset balance, dataset distribution, privacy issues, and accurate labeling. Due to the high control of the data generation user can focus the dataset on a specific task or even generate a Gaussian-distributed dataset to increase the training capabilities of the network. The simplification of the labeling process has an additional benefit because mathematical descriptions set the label and are less influenced by subjectivity, entry errors, or inadequate information [226], summarized in the general problem of label noise [227]. Such an error could be interpreted differently by human or automated labeling tools.

An example would be a very unclear written seven that could even represent a one. The labeling with different human or machine-based labeling with background variations would increase the quality of the label. However, a mathematical description reduces the complexity of the labeling processes and defines a deterministic label.

Synthetic data are essential for training complete systems like autonomous vehicles because capturing all possible scenarios with various sensors and sensor positions is too time intensive and costly. Recorded datasets can also be biased [228], which could come from the manual labeling process by the human. An automatic generation, by mathematical descriptions, of data with the corresponding labels reduces the possibility of bias. There even exists approaches to remove the bias of datasets, mainly on image datasets [229]. A semi-synthetic dataset combines both worlds and can also be seen as data augmentation, where images are rotated, or noise is included to increase the number of samples [230]. One disadvantage of synthetic data is the deviation from the real world, like the noise sensitivity of the algorithm or even the generalization capability of the NN. Often the networks are fine-tuned to improve the performance on the real-world data.

However, due to better control and faster generation, we used a synthetic and semi-synthetic dataset to evaluate the networks. In the following, we describe and analyze the datasets we created during the work.

Table 4.1: Victim radar settings: radar settings of the victim sensor with FMCW modulation. Information based on [40].

Parameter		Value
f_0	initial frequency	77 GHz
Δf	sweep bandwidth	275 MHz
T_{idle}	idle time	$10\,\mu s$
$T_{payload}$	ramp time	$42\,\mu s$
$T_{flyback}$	return time	$2\,\mu s$

4.3.1 Synthetic Dataset

The fully synthetic dataset is generated with an Infineon internal simulator representing the physical functionality of the radar MMIC. This dataset is a part of the following publication [40] and has the advantage that the radar signal consists of the complete signal, which means it also includes the signal between the chirps, the flyback, and the wait time. The simulator utilizes the equations presented in Chapter 2.1.1 and contains measured non-linearity effects within the circuit. The transmit signal is generated in the ramp generator and forwarded to the individual channels and antennas, similar to the block diagram in Figure 2.1. The simulator calculates the intermediate frequency of the superposition of target reflections and interference aggressor signals, as described in Equation 2.12. We apply a band-pass filter of the IF signal to limit the bandwidth and reduce the Alias effects before digital conversion. After the ADC, we use a sampling rate reduction. We depict the victim sensor settings in Table 4.1, where the individual ramp is repeated eight times with the same settings.

Table 4.2: Parameters generated targets: Parameter boundaries of the randomly generated targets. Table modification based on [40].

Parameter		Value Range	
d	Distance	1 m	200 m
v	Velocity	0 m/s	40 m/s
α	Angle of arrival	-50°	+50°
RCS	radar cross-section	-20 dBsm	+80 dBsm

We use randomly generated target scenarios, where we vary the distance, angle-of-arrival, velocity, RCS, and the number of targets. The boundaries of the parameters are shown in Table 4.2. Additionally, we ensure a variety of interference scenarios by randomizing the interference settings, as shown in Table 4.2. We choose the parameter space of the interferer based on the probability of overlapping the frequency courses. Therefore, we mainly generate out-phase interference and reduce the likelihood of in-phase interference, which would need the same ramp settings and timing. Also, important to

Table 4.3: Parameter generated interference: Parameter boundaries of a single aggressor sensor's randomly generated interference settings. The interference effect could be low due to the short duration and low amplitude of the interference, which also occurs in the real world. The table is modified from [40].

Parameter		Value Range	
f_0	initial frequency	76 GHz	78 GHz
Δf	sweep bandwidth	0.2 GHz	1.5 GHz
T_{idle}	idle time	$2\,\mu s$	$12\,\mu s$
$T_{payload}$	ramp time	$40\,\mu s$	$90\,\mu s$
$T_{flyback}$	return time	$1\,\mu s$	$12\,\mu s$
INR	interference-noise-ratio	10 dB	40 dB

mention is that we are limiting the interference to a single aggressor and do not consider multiple ones because the likelihood of multiple interferers is low.

Since we randomize the scenarios, the targets could get very close with a high RCS, which leads to a saturation of the supported voltage range. Therefore, we separate these saturation samples to further investigate the system with an additional class. Another issue is the occurrence of a short interference duration and low amplitudes because its effect on the noise floor is minor and can be neglected. However, these samples are still labeled as interference. Therefore, these samples negatively influence the detection performance by not considering the effect of the interference.

Overall, we generated 460 radar time series without interference and 2266 sequences with random interference, where 51.4% contained interference. Each series contains eight consecutive chirps with the signal during the wait time, as indicated in Figure 2.2. We decided not to further explain the split into the train and test set at this point because one of the methods utilizes only the normality representation. As a label, we generate a binary event for each time step based on the known frequency course of the victim and the aggressor device. Hence, we can verify methods that generate chirp or time-step-based classifications. Thus, we can also use the dataset for pattern classification by splitting it into separated sequences based on the frequency chirp. We follow the same technique as state-of-the-art by only extracting the valid region of the individual chirp and removing the time between chirps and the known settling time of the components.

4.3.2 Semi-Synthetic Dataset

Better coverage of reality achieves a semi-synthetic dataset [231], which combines the advantages of actual measurements and synthetic data generation. Due to the real measurements, it reduces the deviation from reality, and the synthetic superposition of interference patterns allow high flexibility in terms of interference power, temporal position, geographical location, and labeling. Current research focuses within the field of ML on using simulated data to include corner cases and reduce costs and development time. We use the High-Resolution Radar Sensor from Infineon with five MMICs, 16

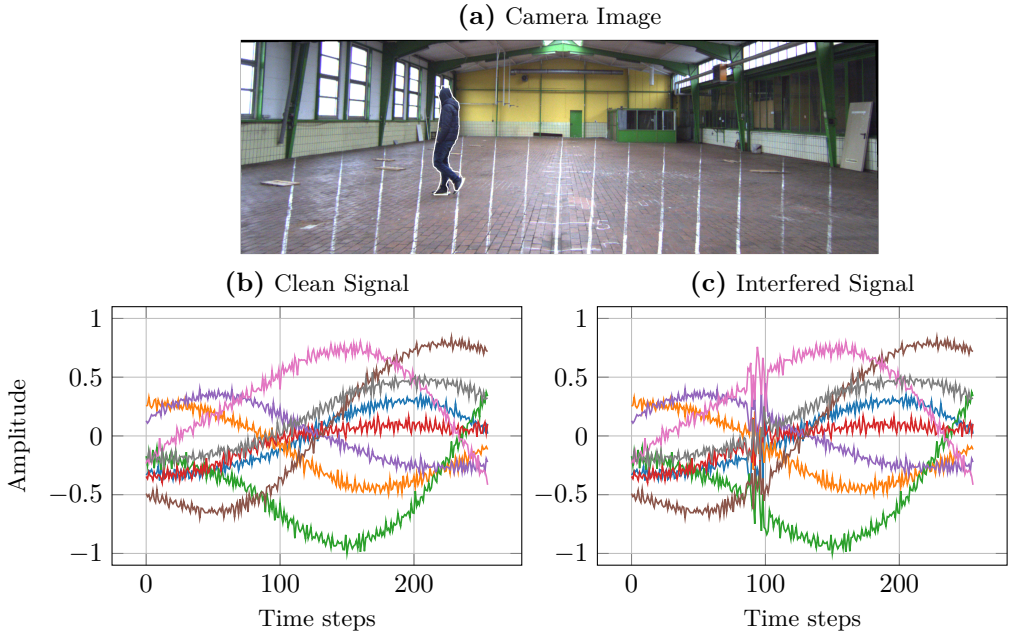

Figure 4.4: Recorded camera and radar data: Example of the radar measurements with the camera (a), clean signals of each channel (b), and the superposition of the clean signal with synthetic interference around time step 100 (c). The individual low-frequency component of each channel appears due to leakage from the transmitter to the receiver, and interference affects all channels simultaneously. We recommend viewing this figure in color. The KI-ASIC project partner OTH Amberg-Weiden recorded the data. [232]

receive, and 8 transmit channels. We implemented our software for the two AURIX microcontrollers[1] that interact with the five MMICs. We use a Python interface on the host side to configure the individual sensors and start the measurement with the transmission of the ADC samples, the Range data, the Range-Doppler data, or the Range-Doppler-Angle data cube via one or two Ethernet ports. The software architecture and program flow of the high-resolution radar software exceed this thesis.

In the research project KI-ASIC, five such high-resolution radar sensors are attached to a BMW x5. Additionally, the project partners mounted one Light Detection and Ranging (LIDAR) sensor and four cameras for recording a dataset with ground-through information. The developed automated labeling pipeline generates the dataset labels, as explained in [232]. In the following experiments, we use the first generated example dataset, recorded inside with targets like a car, pedestrians, and cyclists. In Figure 4.4, we demonstrate one example camera and radar recording of a single pedestrian moving sidewise to the ego-sensor. The left time-domain radar plot demonstrates a clean signal where the low-frequency component affects the transmitter-to-receiver leakage. The right plot shows the same clean signal with a superposition with a synthetic interference

[1]AURIX-family-tx3xx: https://www.infineon.com/cms/de/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc3xx/aurix-family-tc39xxx/

pattern regenerated by our simulator based on the mathematical description in Chapter 2.1.3.

4.4 Assessment of the Normality Predictor

The normality predictor architecture consists of the predictor, outlier threshold, smoothing, and optional outlier mitigation. Every module works for itself and can be replaced by another method. Therefore, we first analyze different predictors on exemplary data and apply the complete outlier detection pipeline on the best-performing one. Additionally, we extend the architecture with zeroing mitigation to present the combined improvements of the detection and mitigation method in terms of SNR. Finally, we use the proposed architecture to show that the method is not limited to outliers from interference.

4.4.1 Predictor Performance

The first experiment focuses on the prediction performance of different predictors. For the training, we used the MSE loss between the target and predicted value with an ADAM optimizer and a learning rate of 0.001. Due to early stopping, the training is halted if there is no improvement in the validation performance for 20 epochs. These settings are used throughout the following experiments.

Nevertheless, we implemented a small toy example before using a predictor to estimate the radar signal. The task is to predict the future sample points of a fixed sinusoidal waveform with unit amplitude, frequency of 250 Hz, and a random phase offset. The series is separated into chunks for the Truncated Back-Propagation Through Time (TBPTT) training as explained in Chapter 4.2.1 and split into a training and validation set. The test set contains the complete series because we do not apply any non-linear effects like noise, and we are not further interested in the deployment of the toy example while providing a comparison between different NN architectures.

The LSTM and GRU architectures consist of a two-layer network with each 25 neurons and a fully connected interconnect without across-layer recurrent connections. The readout layer uses a standard linear layer to convert the high-dimensional space into a one-dimensional representation for each time step. The LSTM layer consists of an input, forget, cell, and output gate with tangents hyperbolic and sigmoid activation functions. A reduced complexity was developed with the GRU layer that uses only three gates, the reset, update, and new gate. Another ANN architecture uses only two dense layers where the input is converted to a higher dimensional representation with ReLU activation and a readout layer to convert it into a single prediction.

The 3rd generation networks with spike transmission use a similar architecture with an encoding, hidden, and readout layer. For the encoding, we either use TC or Current Injection (CJ), which utilizes a population of LIF neurons. The CJ encoding uses a population rate to represent the amplitude-selective neuron where the weights and the threshold defines the tuning curve of the individual neuron; for further information, we refer to [173]. The TC encoder emits spike events for an absolute change between the time steps above the threshold 0.0009. The encoder uses two virtual neurons to indicate

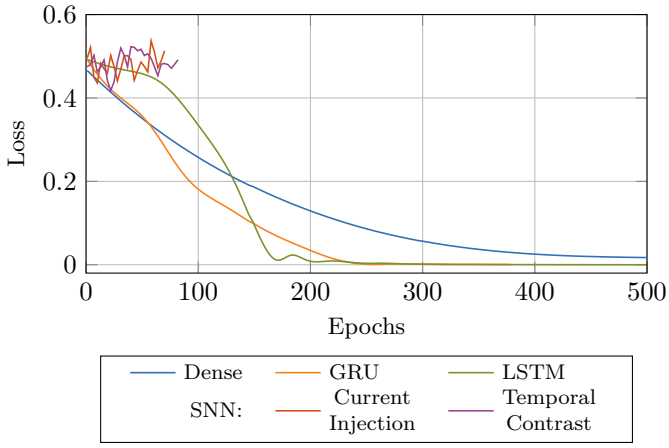

Figure 4.5: Predictor performance across epochs: Comparison of different network architectures to predict a synthetic sinusoidal signal with constant frequency and amplitude. The example indicates how well the networks predict a simple example. Due to early stopping, the number of epochs differs between the architectures. We recommend viewing this figure in color.

the positive and negative change. The hidden layer employs a recurrent connected LIF population with a unit threshold and optimized size and time constants. The number of neurons is 40 and 19, the synaptic time constant is 361.2 and 469.3, and the membrane time constant is 200 and 486.6 for the TC and CJ after the optimization. Important to note that the time constants are unitless in the Norse simulator [223] because the simulation time steps define the unit of the time constants. The readout layer is a linear function of all spike events from the hidden population. We decided against the commonly implemented leaky integrators because the output should represent a continuously changing floating point value instead of an integrator. Therefore, the linear readout corresponds to a trainable version of the TC decoder.

Figure 4.5 shows the learning curve across the epochs. The validation loss, defined as the MSE between the target and the predicted value, indicates how well the network predicts the sinusoidal curve. In this task, a constant predictor that always predictions one value can achieve a loss of 0.5 for always prediction zero, which is the minimum necessary performance of any predictor. We could observe that both SNN architectures have problems solving the task. The TC encoding with a temporal integrator can ideally reconstruct the original signal, and the following network needs to extract the features. However, we could not observe that networks with TC performed as expected during training. They either end up predicting the mean value of the sinusoidal or in a prediction with a sinusoidal shape in the beginning but then increases or decreases, or very unstable predictions around the zero amplitude. Thus, the predictions do not even come close to the sinusoidal target, and as expected, a leaky integrator layer as readout does not further improve the results. Changing the window length that limits the BPTT does not influence the prediction performance. Due to early stopping, the training ends up very

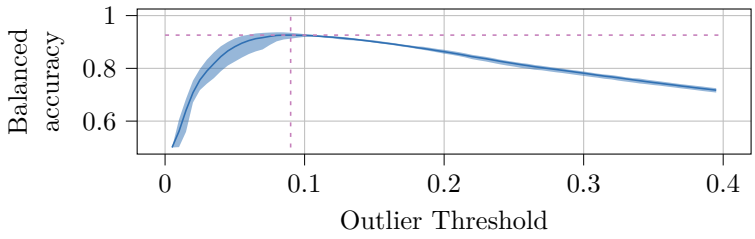

Figure 4.6: Accuracy as a function of the outlier threshold: Correlation between the balanced accuracy and the threshold of the outlier score. Standard deviation reported based on five different random seeds of the LSTM predictor. Indication by dashed horizontal and vertical lines of the highest mean detection rate and the selected threshold for the subsequent experiments.

early, as depicted in Figure 4.5. A further increase in the patient of the early stopping increases the likelihood of overfitting (not important here) but has not improved the prediction.

On the other hand, the performance of the ANN is very stable and fast to achieve. Even a pure dense network without any recurrency improves consequently across the epochs. However, the GRU and LSTM learned faster to predict the sinusoidal signal and reached a very low MSE. Overall, the performance of the LSTM was the best performance with the highest parameter count. The spiking networks used here have a total of $1,158$ and $1,722$ parameters for the current injection and temporal contrast, respectively. The ANN networks consist of 76 (dense), $6,026$ (GRU), and $8,026$ (LSTM) parameters. For the given toy example, the sizes could be further reduced and optimized, which exceeds the experiment's requirements and usage. Due to the low performance of the SNNs, we use for the following analysis only the LSTM network. We have previously published a part of this work [40].

4.4.2 Threshold Estimation

After designing and selecting a suitable predictor to forecast the next sample point of a time series or discrete signal, the outlier scoring module has to be optimized. The basic idea is to differentiate the predicted and the real value by calculating the absolute error. The sample is labeled as an outlier if the error exceeds a predefined static or dynamic threshold. In this work, we use a static threshold function and evaluate the threshold selection based on the LSTM predictor by showing the differences between the accuracy, balanced accuracy, and the F1-score.

The experiment setup combines the LSTM predictor, the outlier score, and the threshold module, as shown in Figure 4.1. Therefore, we trained the predictor five times and calculated the outlier score. Afterward, we evaluated the balanced accuracy for the threshold values between 0.01 and 0.4. We split the radar signal into the individual

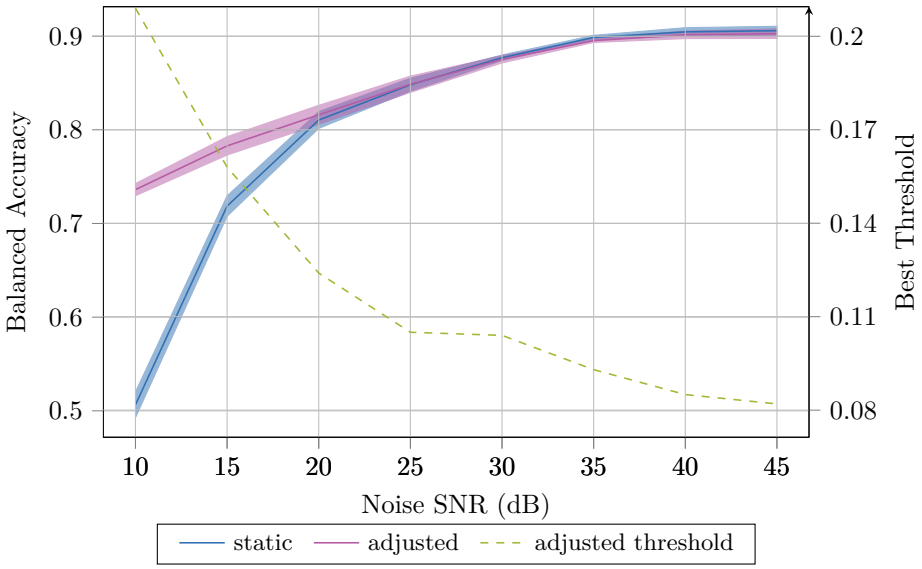

Figure 4.7: Noise robustness of the LSTM predictor: Simulated five pre-trained networks with Gaussian white noise and varying noise levels. The previously defined threshold is kept for the static curve, and the pink line shows the accuracy of the best-selected threshold for each noise level. The dashed line shows the corresponding mean threshold of the simulated networks. The higher noise level increases the necessary threshold value. We recommend viewing this figure in color.

chirps and evaluated if any sample was detected as an outlier to reduce the effect of not precisely detected outlier samples.

Figure 4.6 shows the experiment results with different thresholds. As expected, the accuracy increases until the best-balanced accuracy due to a balance of the false and true positive detections. Afterward, the performance decreases because the likelihood of false negatives increases. We also see that the standard deviation between the networks is neglectable but higher for smaller thresholds. We selected the threshold of 0.09 based on the best mean balanced accuracy for the following evaluations. Furthermore, we discovered the same threshold by the evaluation of the F1-score [40].

Consequently, we correctly classified 93.6% of chirps and 67.0% of the sample as labeled. The high difference is also detectable in the precision of the outlier class, which improves from 0.39 to 0.86 due to the different interpretations of the detections. These results are consistent with the observations that it is impossible to detect all corrupted samples because the signal's amplitude is within the predicted range by chance and therefore affects the detected outliers. One possibility to improve the localization is the smoothing already proposed in our publication [40], which was implemented by a simple counter but could also be realized by a convolution.

4.4.3 Noise Robustness

Another aspect of the predictor is the sensitivity towards noise that can occur due to temperature changes, disturbance in the power supply, or other effects. Hence, we investigate the performance reduction due to increased adaptive white Gaussian noise. We sweep the SNR between 10 dB and 45 dB and calculate an additional noise based on the estimated radar signal power of the sum of squared sample points. We simulate five pre-trained LSTM networks with randomly generated noise for actual deployment and report the chirp-based balanced accuracy. Additionally, we analyze whether the redefinition of the threshold improves the noise robustness.

As expected, we can observe a clear accuracy reduction with the noise level increase as depicted in Figure 4.7. Below a SNR of 40 dB, the balanced accuracy drops below 90% due to the noise level and continues to degrade. Based on data without additional noise, the static or predefined threshold provides good performance for the low-noise outlier detections. However, with a higher noise level, the performance drops are immense, and the system is not usable anymore. With a SNR of 10 dB, the performance reduces to 50% accuracy, but with a redefinition of the threshold, the system can reach an accuracy of 73%. Due to the noise, the amplitude and the expected outlier score increase, which then affects the outlier detection. This scenario can be mitigated by readjusting the threshold value, but in actual system deployment, this can only happen under controlled circumstances; otherwise, the number of missed classifications increases heavily.

4.5 Evaluation of the Pattern Classification

In this work, we use the radar interference detection task as an example to answer the research questions about the architecture design and selection of encoding scheme for radar signals. Therefore, as explained previously, we implemented different architectures based on traditional and spiking networks and evaluated the methods with the semi-synthetic dataset.

4.5.1 Baseline Methods

Before evaluating the performance of the NNs, we adjust the settings of the baseline methods. We search for suitable parameter settings for the thr-simple, simple static threshold, and thr-windowed across the standard deviation of a moving window because the algorithms are not trained. Therefore, it is valuable to visualize the relation of the true positive and false positive rates for various threshold candidates, as shown in Figure 4.8. Here, the positive class represents the interference where we want to achieve a high true positive rate with a low false positive rate for good model performance. A classifier that cannot separate the classes predicts either a random or a constant class corresponding to the diagonal dashed line. We have selected the threshold value based on the best relation of the true and false positive rates. Figure 4.8 additionally shows the higher performance of the windowed threshold, where we still see room for improvement. The thr-simple has a validation accuracy of 56.3% and thr-windowed of 87.5%.

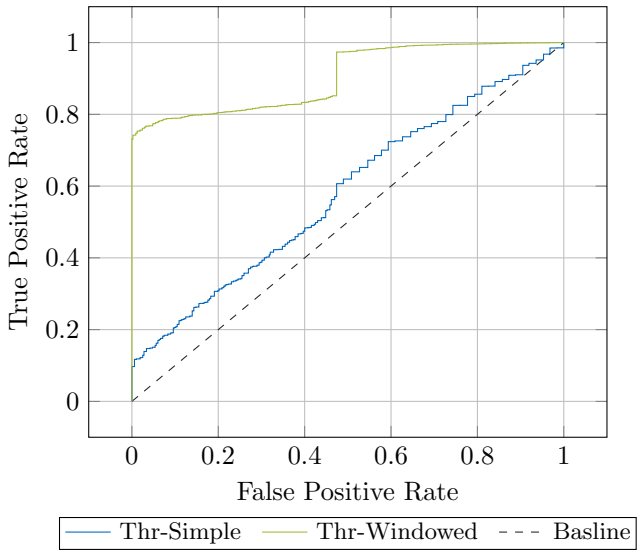

Figure 4.8: ROC of threshold methods: Illustration of the relation between the true positive and false positive rate for the two threshold-based methods. The curve indicates the trade-off between high detection and low false detection rate. Therefore, it is a purpose to achieve the left upper corner. We recommend viewing this figure in color.

Another baseline method is the thr-iterative proposed by [208], where the threshold is iteratively reduced to distinguish the strongest signal from the interference. Also, we used a hyperparameter search to find a suitable parameter setting for the training set. We used a threshold factor $k = 1.887$ and a minimum difference between the iterative thresholds of $\Delta 0.965$ and achieved a test accuracy of 58.2%. All baseline methods do not perform as expected on the dataset because the radar signal has a low-frequency component due to the leakage of the transmit signal into the receiver. Therefore, the level of interference is within the signal range. Higher interference signals would lead to saturation, which can already be detected in today's sensors. The thr-windowed is less sensitive because of the windowed standard deviation that mitigates the offset of the low-frequency component and is therefore seen as a baseline for the NN methods.

4.5.2 Architecture Comparison

This chapter presents a fundamental comparison of various network architectures. Before diving into optimizations for individual networks, this initial assessment will lay the groundwork for our analysis.

Unlike the baseline methods, the NN-inspired algorithms have a complex design space, and some parameters, especially of the SNNs, are not trainable. Thus, we use a hyperparameter search to define the necessary frequencies and the neuron threshold of the R&F population with the hybrid R&F_b-CNN network because the CNN runs natively on the GPU with a high performance [233]. Afterward, we investigate the minimum parameters for the extracted encoding frequencies within the R&F_b-LIF architecture

representing an end-to-end SNN. The other network architectures are aligned such that the number of parameters is around $3,600$ for a fair comparison. Especially edge devices have limited resources in memory and processing, and every architecture has different restrictions and architectural properties. Without this limitation, an accuracy of 98% can be achieved by roughly 16 million parameters with the STFT-CNN network, which exceeds, by far, the memory of an edge device. The naming of the architectures follows the order shown in Figure 4.3 with the differentiation that b and g mean binary and graded spike function.

In all parameter selection experiments, we use the optuna [234] framework for the hyperparameter search using a Tree-Structured Parzen Estimator [235]. As previously explained, all networks use the same learning routine, optimizer, and loss definitions. The hyperparameter search focuses on the network architecture, like the number of hidden neurons, the kernel size, the learning rate, and the number of encoding neurons for the R&F neurons. The selected architectures are detailed in Appendix A. Table 4.4 depicts the loss, accuracy, average spike count per inference, and the parameter count of the selected classical, CNN, LSTM, and SNN architectures. We report each architecture's mean and standard deviation by five simulations to reduce the effect of random weight initialization. The traditional methods are only verified ones because there exists no parameter that is initialized randomly.

Furthermore, the networks with an encoding threshold are also optimized like the current injection networks, where a smaller threshold value showed a better performance. However, there is a high correlation between the threshold and the weight initialization because the input weights scale affects the threshold value. Also, the TC method uses a threshold to differentiate the temporal changes between an important feature and noise, which we selected based on an extensive parameter search.

As previously analyzed, the classical methods utilize a very low parameter count but also have a low performance compared to the other methods, which are split into the three categories of traditional NN, hybrid, a combination of traditional and spiking methods, and the pure SNN approaches. A random classifier that always predicts randomly or always the same class would reach an accuracy of 50% because of the balanced binary classification task. Therefore, the performance of the LSTM is very low and demonstrates that it cannot extract the important features from the time series. Interestingly, both LSTM networks show the same loss and accuracy, but a slight neglectable difference demonstrates the problems during the training of the networks. Additionally, we observed that an LSTM architecture with $483,032$ parameters could solve the task with comparable accuracy, but the immense parameter count is unsuitable. The parameters contain all necessary weights, thresholds, encoding frequencies, and important variables to deploy the network without considering the buffering of temporal data. We also included the number of coefficients for the STFT processing that are most likely stored to reduce the processing latency. In general, more parameters increase the probability of good network performance but also increase the complexity.

The two CNN architectures without and with encoding already show an improvement in the classification with limited parameters. The STFT-encoded CNN has the highest performance of all networks reported in Table 4.4. However, the CNN methods have one

Table 4.4: Network hyperparameter search: Evaluation of different network architectures and encoding methods for solving the interference classification task. We aligned the architectures based on the total number of parameters around $3,600$. Due to the different architectural properties, we slightly vary between the architectures. The dataset has a balanced class distribution; therefore, we use the accuracy metric and the average total spike count per inference.

Name	**Validation Performance**		**Spike Count**	**Number of Parameters**
	Loss	Accuracy		
		Traditional		
thr-simple		0.563		1
thr-windowed		0.875		2
thr-iterative		0.582		3
		ANN		
LSTM	0.693 ± 0.0	0.5 ± 0.0		$3,728$
LSTM-Dense	0.693 ± 0.0	0.5 ± 0.0		$3,530$
CNN-Dense	0.265 ± 0.032	0.897 ± 0.016		$3,602$
STFT-CNN-Dense	0.1 ± 0.02	0.962 ± 0.007		$3,650$
		Hybrid		
R&F_g-CNN	0.144 ± 0.006	0.947 ± 0.004		$3,637$
R&F_b-CNN	0.118 ± 0.008	0.958 ± 0.004		$3,637$
STFT_b-LIF	0.693 ± 0.0003	0.829 ± 0.017	1.6 ± 1.2	$3,622$
STFT_cj-LIF	0.693 ± 0.0001	0.595 ± 0.072	2.72 ± 1.43	$3,622$
		SNN		
Cj-LIF	0.693 ± 0.0002	0.514 ± 0.005	$4,100 \pm 816$	$3,600$
TC-LIF	0.437 ± 0.039	0.79 ± 0.049	$2,391 \pm 584$	$3,597$
R&F_org-LIF	0.122 ± 0.0047	0.96 ± 0.003	810 ± 132	$3,585$
R&F_b-LIF	0.126 ± 0.0135	0.955 ± 0.006	896 ± 88	$3,585$
R&F_g-LIF	0.225 ± 0.1244	0.922 ± 0.045	602 ± 170	$3,585$
R&F_b-fLIF	0.131 ± 0.008	0.954 ± 0.004	318 ± 36	$3,616$
R&F_g-fLIF	0.227 ± 0.011	0.914 ± 0.004	195 ± 43	$3,616$

big drawback because at least a portion of the input time series needs to be buffered to process the different channels compared to all other methods. The preprocessing with the STFT improves the performance from 89.7% (CNN-Dense) to 96.2%. While the parameters are similar, the network architecture differentiates because the STFT transforms the series into a spatio-temporal representation, where some information is removed due to redundancy, like the overlap of consecutive FFT windows. Thus, the reduced network input size directly influences the number of parameters, and more channels with a lower stride are applicable. This indicates the importance of a suitable encoding method that extracts important features and removes unnecessary information.

Table 4.4 shows different spiking networks with varied encoding methods. The famous current injection (Cj) encoding method shows a low performance with a high parameter count compared with the LSTMs. As expected, the performance of the TC encoding is higher than the current injection by 28% percentage points. Consequently, it demonstrates that the temporal features of the series can be better extracted by the TC encoding, which could be explained by the fact that the current injection focuses mainly on the signal's amplitude instead of the temporal changes. Temporal information is also encoded by the frequency-time dependent spikes of the R&F encoding, and in combination with a population of LIF neurons, the accuracy increases to 96%.

Interestingly the original encoding architecture by Izhikevich [115] has the highest accuracy but is comparable to the phase-dependent spike mechanism. The networks with only forward connections perform similarly by using fewer spike events. One possible explanation is the inherent recurrency of the neuron itself, and the neuron state already carries information about the history and provides additional information. On the contrary to the higher information density of the graded R&F encoding, the accuracy of the networks with and without recurrent connections decreases.

Another observation is the correlation between the average spike count per inference of the hidden LIF population and the accuracy of the graded spikes. The graded mechanism always leads to a lower spike count with a lower performance. The directly connected LIF population has the highest spike count of $4,099.69$, that most likely does not end up in an efficient architecture because in each time step, 16 spikes exist, and each spike increases the number of operations. Otherwise, in the temporal contrast approach, the spike count is reduced but still high compared to the architectures with R&F encoding.

Furthermore, we investigated hybrid architectures, which are combinations of STFT or R&F encoding with spiking or traditional NN, respectively. In the example of the STFT encoding, we use the current injection and a simple binarization method by defining an event when the amplitude is higher than the moving mean, similar to [44]. During training on the current injection approach, we could observe vanishing gradients due to the reduced time steps by the STFT. The low performance is also represented by the very sparse spike events of the network. With the binarization method, the network reaches an accuracy of 82.9%. The reduced number of simulation steps decreases the performance of both approaches and only by increasing the resolution a higher performance can be reached. Combining the R&F encoding with a CNN classification demonstrates that the binary events carry enough information for the classification and shows that graded spikes do not contribute more to the classification of the interference detection task. However,

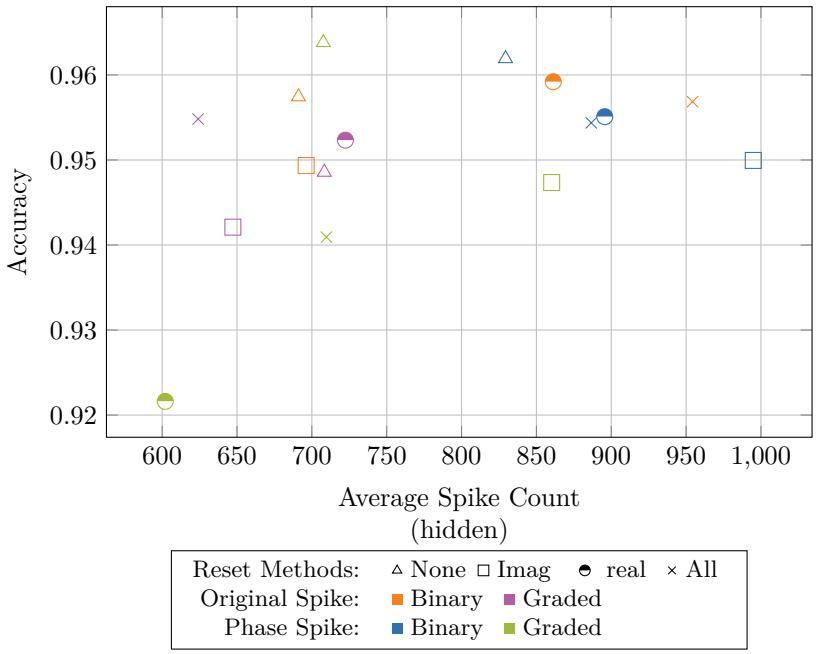

Figure 4.9: Evaluation of reset and spike methods: Comparison of different reset methods of the R&F (markers) and the binary and graded spike activation. We use the mean accuracy against the mean average spike count per inference of the hidden population of LIF neurons across five runs. We recommend viewing this figure in color.

all networks with the graded R&F encoding have a lower performance. Therefore, we will further investigate the effect of various reset and spike mechanisms of the R&F neuron and focus our evaluation and optimization on the pure spiking variants.

4.5.3 Reset and Spike Mechanism

As mentioned in Chapter 3, using graded instead of binary spike events is possible. For R&F neurons, the grade carries the magnitude of the frequency component in relation to the previous spike events. Also, the reset function after an output event can be implemented in different versions. It is possible to not reset the neuron at all (none) and reset a single or both states (imaginary, real, all), which leads to four variants. Another possibility is to reset the states by a specific value instead of a full reset, which will not be further investigated here.

To analyze the differences between the spike and reset functions, we use the R&F-LIF network architectures as a base and alternate the reset and spike functions while simulating each network for five iterations to reduce the probability of an outlier. We focus our evaluation on the mean accuracy and average spike count of the hidden population per inference, that is, the mean across all samples in the dataset.

Figure 4.9 summarizes the results of the different combinations, where the markers correspond to the reset function and the color encodes the spike function based on the

original and phase-dependent spike method. For clarity, we only report the mean of the networks here and do not further visualize the standard deviation. The different accuracies of the graded and binary spike mechanisms are obvious because the networks with binary spikes reach higher accuracies. However, the phase-dependent graded spike with no reset is an outlier because of the highest accuracy with a mediate spike count. On the other hand, the same approach with a reset of the real component leads to the lowest accuracy with the lowest number of spikes. Another interesting observation is that the number of spikes is lower for most graded spikes than for the binary spikes, which could indicate more information within a single spike event from the input layer. The spike count has to be seen in relation to the maximum possible spike events, which are defined by the number of neurons and time steps. In this experiment, all the networks have the same architecture; therefore, a maximum of $3,584$ spikes is possible. Whether the spike count reaches the maximum number of spike events, this architecture is unsuitable for neuromorphic processing since a low spike count is desired for event-driven processing. Important to highlight is here the difference between a synaptic and a spike event. A spike event is triggered when the threshold of a neuron is reached, and a synaptic event depends on the implementation but means either a weight lookup or multiplication. Thus, a single neuron output can be connected to x other neurons, and therefore, the single spike event needs to be multiplied by x to extract the number of synaptic events. We use the spike events here, but the synaptic events can be calculated for a fully connected network.

Also, the spike count of the encoding layer changes for the different reset mechanisms without any effect by the graded or binary spike function. As expected, the highest spike count of $2,203$ appears during no reset, and a slight improvement is achievable by using the imaginary reset that leads to $1,520$ spikes per inference. We observed only 295 spike events for the real and all reset methods.

The rest of the imaginary component has the lowest performance and a high spike count because of the direct connection of the input signal to the real component. Imagine the phase diagram of an oscillating input and a reset of the imaginary part to zero when the threshold is reached. Therefore, the phase rotation is limited to $90°$ because $Im(z[t])$ will always be between 0 and the threshold. This limitation leads to an infinite spike response with a linear interspike interval while the oscillation exists. The use of graded spikes introduces another effect by the constantly increasing magnitude of the phase rotation the grade increases. This could explain why these networks have slightly lower performance in the given task than the binary ones. Interestingly, the real and all reset approach led to moderate performance, but here, the graded version has a lower spike count of the hidden population. The exact spike count of the encoding layer for the real and all reset indicates the higher importance of the real component because without resetting, the real component explodes.

Overall, the difference between the shown variants is low, and most of these techniques end up in a well-performing network. However, a wise selection of the encoding method provides a lower spike count in the encoding and hidden population with a good performance. In the given task, the phase-dependent spike mechanism does not

carry additional important information because interference means a temporal frequency change compared to the static frequency of the targets.

4.5.4 Membrane Time Constant

In this work, the network consists of two time constants the membrane and the synaptic time constant. The membrane time constant defines the time for the decreasing potential towards the neuron's resting potential. Similarly, the synaptic time constant limits the duration of the current flow from the synapse to the neuron input. In the implementation, we use the exponential time constants that depend on the capacity C and resistance R and can be calculated by $\tau = RC$ [236]. Norse uses internally the inverted time constant to reduce the number of calculations, but we report the non-inverted τ here. A high time constant means more historical information is kept in the membrane potential, but this also influences the neuron's activity because it takes longer to charge the neuron membrane potential, reducing the temporal resolution. Additionally, the recurrent connection contributes to the information from historical events that make a high time constant unnecessary.

In the following experiment, we investigate the time constant of each neuron and synapse layer in the R&F-LIF networks with binary spike events and real reset. Therefore, we use again the Optuna framework to analyze the correlation between the time constant parameters by running 200 trials. All the neurons and synapses in the network use the same parameters to reduce complexity, and the time constant is swept in the range from $1.0 \cdot 10^{-10}$ to $1.0 \cdot 10^{-2}$.

The first observation tries to find an answer to the question of which time constant parameter has the highest effect on the network's performance. Therefore, we use the hyperparameter assessing method proposed by [237]. We use τ_{m} and τ_{s} as the membrane and synapse time constant. The τ_m of the hidden population has by far the highest importance of 91.6%, and in both layers, the τ_s has a lower effect with 0.2% for the hidden and 2.4% for the readout. Consequently, the readout τ_m importance is 5.8%. This observation contradicts our expectation that the readout time constant is more important than in the hidden population. However, the results demonstrate that selecting the hidden population parameters is essential.

Figure 4.10 depicts the correlation of the hidden τ_{m} with the other time constants. The color of the contour plot indicates the accuracy where darker is better, and the black circles highlight the tested networks. From the distribution of the simulated networks, we can see the direction of the parameter selection algorithm. First, most simulations are performed for lower τ values because of the performance. Additionally, we observe that τ_{s} of the readout and hidden layer have a great variety compared to τ_m. Therefore, the selection of the synaptic time constant is less relevant for the system performance. However, for the τ_{m} of the readout, it is also better to select a smaller value.

Additionally, we analyzed the number of spikes in relation to the τ values but could observe the connection between the spike count and the accuracy. However, the experiment proves that the initially selected τ parameters are already in a good range because the hidden layer time constants are around 10^{-7}, and the readout τ is at 10^{-4}. In the

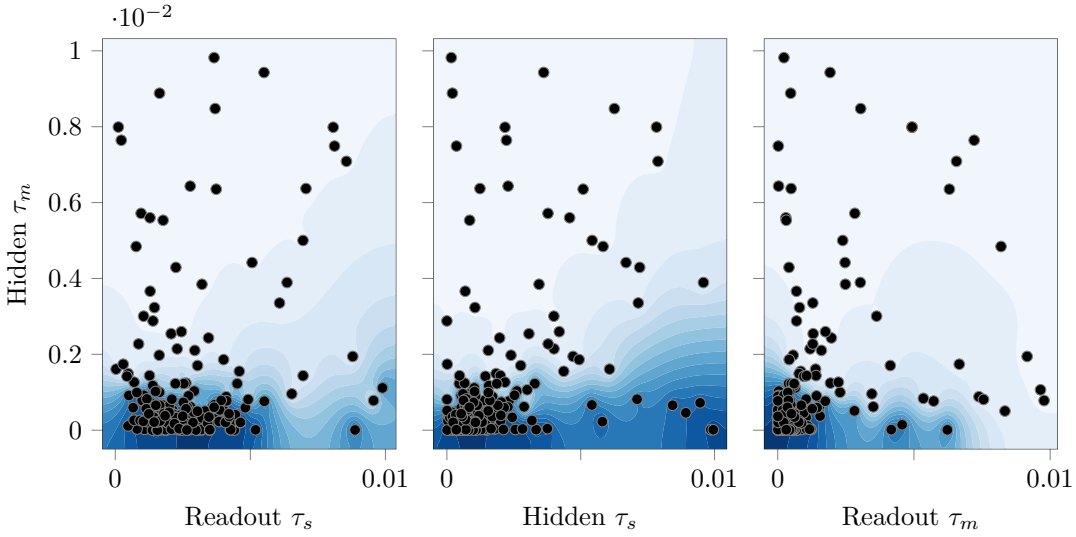

Figure 4.10: Time constants correlation: Relationship of the membrane and synaptic time constants of the hidden and readout layer in the R&F_b-LIF network. The darker blue indicates a higher accuracy, and the dots represent the exact simulated networks. In general, networks with a higher accuracy have lower τ values.

following experiments, we will keep the selected time constants as reported in the detailed network description in Appendix A.

4.5.5 Spike Regularization

The original idea of regularization aims to increase the generalization of the network and reduce the likelihood of overfitting. In the following chapter, we will investigate the influence of the spike regularization of the R&F networks and the contribution to the event sparsity. We use the graded, binary, and binary forward-only network architectures to analyze the spike regularization. Therefore, we train the network with varying spike regularization factors λ and analyze the accuracy and average spike count per inference. Due to the training problems with the encoding layer, we limit the spike regularization to the population of LIF neurons and compare only the average spike count of the hidden layer because the spike events of the encoding are the same for each architecture.

As expected, Figure 4.11 depicts a close correlation between the accuracy and the number of spikes. The decreasing spike count directly reduces the network performance. The first observation is the initial spike count, where the recurrent variants need more spike events than the forward-only network with similar performance. Important to remember is that we have assumed a parameter count of $3,600$ for all architecture, and due to the removed recurrency, more neurons are used. Also, we can separate the accuracy into three regions: 1) the slight performance degradation, 2) the transition or unstable region, and 3) the silent region. For small spike regularization factors below 0.01 the performance decreases slightly but with an exponential reduction of the spike count.

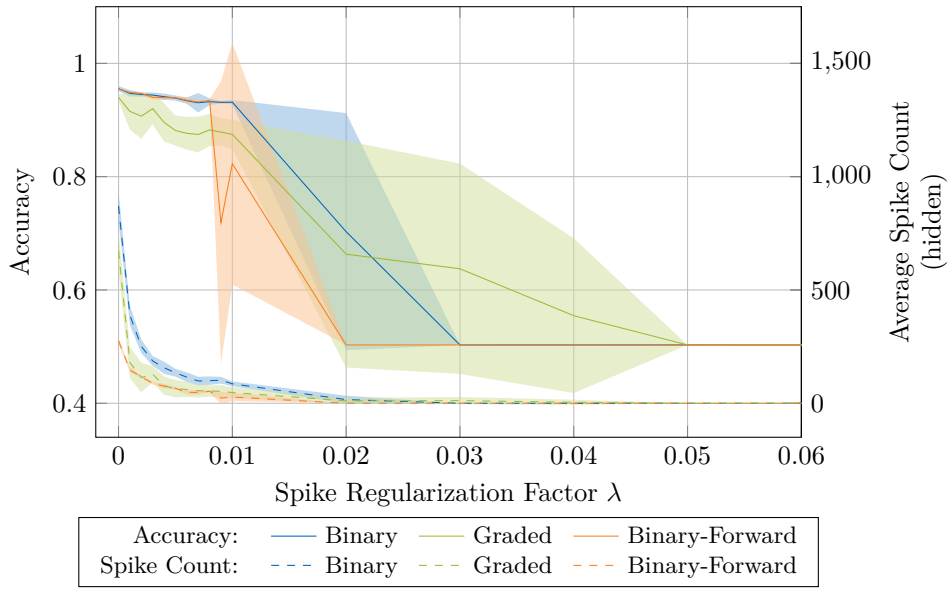

Figure 4.11: Effect of spike regularization: Correlation between the average number of spikes per inference and the spike regularization factor λ. The spike count influences the performance of the interference detection task. We recommend viewing this figure in color.

It is advisable to reduce the performance for a higher energy efficiency depending on the application. The second region is driven by a high instability of trained networks, as depicted by the high standard deviation, and the performance highly depends on the removed synaptic connections. This regime should be avoided because, of luck, it is possible to get a working network, but we expect a terrible generalization due to the low spike response. No apparent regularization factor describes the transition to the third region, as shown by the forward and the graded recurrent networks. The third region starts at 0.02 and 0.05, respectively. However, the third region consists of completely useless networks because, during training, the punishment for a false detection is less than the spike count. Therefore, the trained networks do not emit spike events and consistently predict the normal class, which means an accuracy of 50%. This experiment has demonstrated how carefully the spike regularization factor has to be selected depending on the application's specific loss definition. In our case, we decided to use in the following experiments a spike regularization factor of 0.003 to further reduce the number of spike events without massive accuracy degradation.

Figure 4.12 demonstrates the spike response of the hidden LIF population with the binary input spikes from the R&F encoding on a randomly selected sample. We only differ the spike regularization from 0.000 to 0.003, respectively, shown in (a) and (b). The selected scenario consists of multiple interference patterns around time step 40 to 60, 140 to 160, and a very short one at around 240. In plot (a), the spike train of neuron 10 could be directly related to the interference because the spikes occur around the interference pattern with a small-time shift due to the temporal integration of the neuron. Also, a

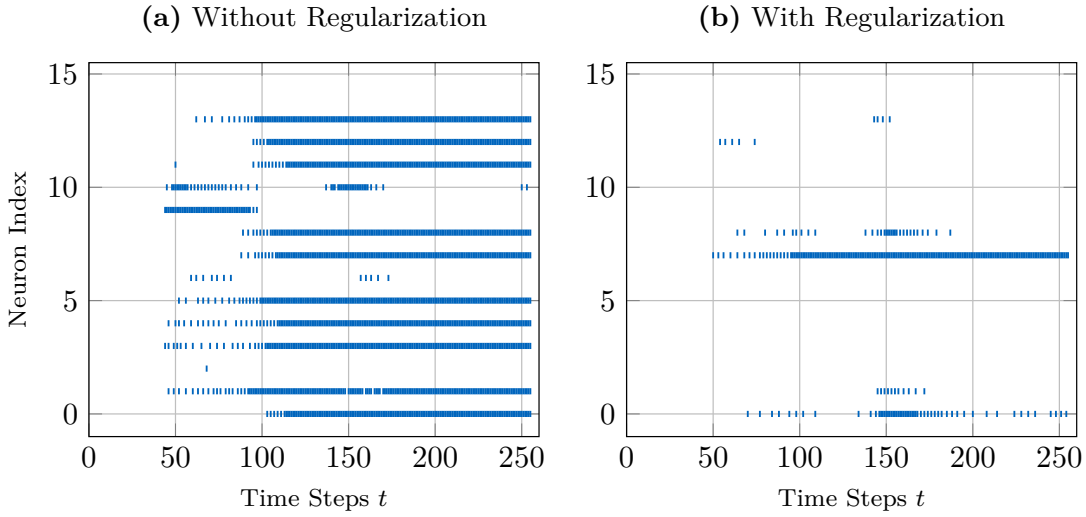

Figure 4.12: Spike response with spike regularization: The spike events of the hidden population in the R&F_b-LIF with binary spike events. (a) does not use any spike regularization, and (b) uses a factor of 0.003. Interference occurs around time step 150 and triggers a high spike response rate.

similar behavior can be observed in plot (b), but it is not as good differentiable. The neuron index 12 and 13 only fire for the first interference patterns that could be a reason for the removed synaptic connections but for the given task, such detection is sufficient. However, plot (a) indicates that the exact position of the interference can be detected by analyzing the spike train differently or training the network to detect the exact position. Another observation is that no spike events occur in the beginning because the R&F neurons are initialized with zeros and, therefore, need a settling time which negatively influences the pattern detection at the beginning of the series. Such a scenario would not happen in a continuous implementation. One remark is that we cannot compare the neuron indices because the focus of the individual neuron will always change by retraining the architecture. However, evaluating the average number of spikes per inference and comparing the spike response, we have highlighted the positive effect of small-weighted spike regularization. Therefore, we can recommend selecting the spike regularization factor based on the expected target loss because if we target a loss of 0.1, then the spike regularization should have a minor influence on the overall loss function.

4.5.6 Network Sparsity

Spike regularization increases the sparsity of temporal events and reduces the number of operations across the time dimension. However, spike regularization does affect the memory footprint because the network parameters and states are stored in the local memory. In the case of a sparser activity, the necessary memory accesses are reduced, but on edge devices, the memory size is already a limitation. Therefore, pruning can

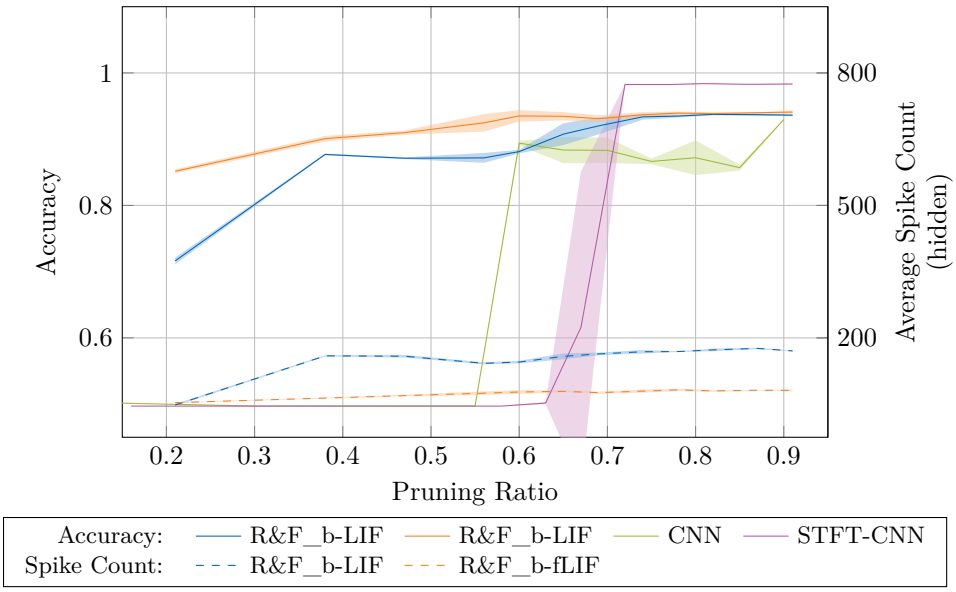

Figure 4.13: Network pruning: Evaluation of the correlation between the pruning ratio, the accuracy, and the average spike count per inference with the hidden layer. A higher pruning ratio means fewer connections are removed. We recommend viewing this figure in color.

keep the network performance by reducing the network size. The main idea is to remove unnecessary synaptic connections, as described in Chapter 2.2.5.

As part of the magnitude-based pruning [167], we randomly select the weights with the lowest L1 norm and remove them by setting the weights to zero. The pruning ratio describes the percentage of active connections and provides the number of pruned connections for the random selection. After the pruning, the network is retrained to mitigate the synaptic loss. In the following experiment, we prune the R&F architecture with a reset of the real component and the binary spike encoding to investigate the correlation between the performance and the pruning ratio. Therefore, we trained the network with early stopping and applied different pruning ratios with a followed network fine-tuning. However, we do not analyze the effect of an iterative pruning technique.

In Figure 4.13, we show the correlation between the pruning ratio, the accuracy, and the average spike count per inference of the hidden population. We prune the hidden population's input and recurrent weights and the readout layer's input weights. We do not further reduce the input weights of the encoding layer because this is basically neuron pruning, which we already did during the initial architecture search. As we can see, the network spike activity is within the expected range for a spike regularization of 0.003 by a pruning ratio of 0.9. In case of no spike regularization, the hidden population emits more than 800 spike events.

Also, we demonstrate the pruning of the CNN architectures, where we prune all weights of each layer except the encoding. The STFT-encoded CNN architecture has the highest

accuracy and is constant until a pruning ratio of 0.7 and then drops rapidly into the random classifier regime. The pure CNN has a lower initial accuracy but drops later into a region with no functionality. From this observation, we can separate the pruning into three regions: 1) The pruning has minor effects until a pruning ratio of 0.72, and 2) the accuracy drops, and the standard deviation increases. 3) starting roughly at 0.65, we reach the silent stage, where the network cannot perform the task anymore and only detects the normal class. We observed the same three regions with the non-spike regulated SNN architectures. These three regions also appear in the weight distribution of the individual layers. The connection between the hidden population and the readout has a flattened weight distribution compared to the input-to-hidden and hidden-to-hidden connections. The network compensates the removed weights with a shift of the active weights, but the number of synaptic connections is too small at a certain pruning level. In general, we observed with and without pruning that the mean of the hidden weights is shifted into the negative region. Figure 4.13 shows only the regularization networks for clarity. Interestingly, the performance drop is low compared to the CNN and the unregularized network. However, the recurrently connected networks drop at a pruning ratio of 0.7, and the forward-connected architecture drops at 0.6. The performance of the forward network is still above 90% for a pruning factor of 0.5. This means half of the synaptic connections can be removed. Thus, the parameter count of $3,616$ reduces to $1,808$. A further reduction also decreases performance and could negatively influence the training. Another observation is the low changes in the spike count, where the number of spike events also observes the drop in accuracy. During 256 time steps, the hidden population reduces the spike count from 170 to 160, which is neglectable compared to the effect of spike regularization. The improvement of the spike count is even more minor for the forward-connected network.

Thus, we have proven the lottery ticket hypotheses by demonstrating the existence of subnetworks within the initially defined architectures on the same task. Although the traditional ANNs can perform better, they are more sensitive to removing synaptic connections than the SNN.

A balance of excitatory and inhibitory synaptic connections in biological systems exists to increase noise robustness [238], [239]. We have discovered a shift towards inhibitory connections across all layers in our networks. For example, the R&F network with binary spike events and recurrent connections uses $1,596$ inhibitory and 935 excitatory connections for the input-to-hidden, hidden-to-hidden, and hidden-to-readout weights.

Additionally, we analyzed the effect of weight regularization that forces the weights closer to zero to reduce the possibility of overfitting [240]. We discovered the same effect as with the spike regularization. For a high factor, the performance stabilizes at 50% because the reward of a small weight is higher than the correct classification. The accuracy of the SNN drops to 2% with a weight regularization factor of 0.001. We could not observe any benefits of weight regularization. Therefore, we do not apply it in the following experiments.

4.5.7 Network's Noise Robustness

A further often-mentioned advantage of SNNs is the inherent noise robustness compared to traditional ANNs. [241] demonstrated the noise sensitivity of a non-spiking and spiking network on estimating the trajectory based on a radar sensor. Their findings of higher robustness of SNNs are argued by the difference in the activation function because the step-activation of spiking networks is less sensitive to noise than sigmoid or ReLU functions. For analysis, they used the input data with additional Gaussian white noise or impulse noise to train the network. Their findings indicate that SNNs are more robust. A controversial result is the increasing test loss of the non-spiking architectures, which usually indicates overfitting and reduces the networks' comparability. However, the noise robustness introduced by the Heaviside step function is not investigated in terms of the encoding method. Therefore, we will analyze different R&F and TC encoding techniques and compare them with the previously explained CNN architectures.

Noise is an expected component of real-world implementations, and for automotive applications, it is necessary to investigate side effects that can degrade the system's reliability. Therefore, we selected six networks with different encoding properties based on ANNs and SNNs. The five versions of the pre-trained networks with spike regularization of 0.003 and pruning of 0.8 are tested under noisy conditions without retraining. Important to note is that the TC network does not use spike regularization or pruning because the accuracy is already below the baseline methods. The noisy dataset is a superposition of the validation data and Gaussian white noise with different target SNR values. The noise is drawn from a uniform distribution with zero mean, where we calculate the noise power based on the expected target SNR, which varies from 100 dB to 5 dB. We define the signal power as $10\log_{10}(\sum_{t=0}^{T} x[t]^2)$ for each individual sample. Important to understand is the fact that the original dataset already contains noise from the real measurements. Thus, the noisy data are expected to be at the border or outside the training distribution and therefore indicate the reaction to untrained environmental effects, such as the rapid temperature increase or the misfunction of electronic components within the sensor or the neuromorphic circuit.

The results of the noise simulations are depicted in Figure 4.14 with the selected networks. The CNN architectures show the highest stabilities in regard to the noise level because the accuracy decreases below 35 dB, which we observed in the pure CNN and by the network with STFT encoding. The STFT-encoded network performs better but loses earlier the classification functionality than the pure CNN. The CNN architectures are inspired by the image processing field, and there it is well-known that specific kernels can sharpen, blur, or even denoise images [242]. Therefore, the input layer can act like a filter of the Gaussian White noise and can suppress the occurrence of a zero mean noise by averaging across multiple features. The TC network has the lowest initial performance but has the most robust behavior. At a target SNR of 20 dB, the network still predicts 60% right compared to most other architectures with 50%. One possible explanation is that the TC is the only architecture that directly converts the amplitude information into spikes based on the changes. Because the original training data already contains noise, the encoded spikes forward the noisy events to the hidden population. Thus, the

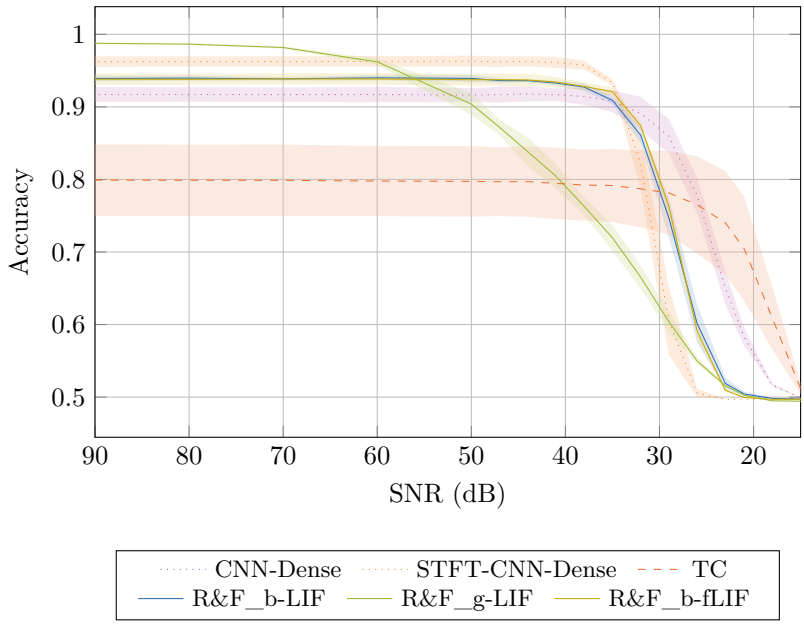

Figure 4.14: Performance degradation through noisy input: Relation of the network performance to an increased noise level without retraining the network. A network with high accuracy at low SNR is desired. The R&F networks use a spike regularization of 0.003 and a pruning factor of 0.8 like the CNNs but not the TC and graded architectures. We recommend viewing this figure in color.

hidden and readout population is trained with higher robustness to the spike events, which correlates with the overall reduced performance. One example of such noise is the TX leakage that introduces a low-frequency component and implements a non-linear shift across time. This directly influences the TC encoding and makes it more robust.

Another interesting observation is the equal behavior of the R&F encoding with binary spikes. Although, we have trained both architectures with spike regularization and pruning, where we expected a higher variation. However, this demonstrates two properties: first, the Gaussian White Noise leads to a wideband noise distribution, and the single frequencies are less affected than the overall amplitude, and secondly, the noise robustness is mainly driven by the encoding approach and the defined threshold value. A lower threshold also means a higher sensitivity toward noise. Interestingly, the R&F encoding with binary spikes is from the accuracy perspective below the STFT-CNN but above the CNN and from a noise robustness view between the two network variants, where the CNN is less sensitive to noise. For curiosity, we have also evaluated the R&F encoding with graded spikes with the highest accuracy of all networks, but no complexity reduction techniques were applied. The higher accuracy was unexpected compared to the simulations without noise, indicating that a slight Gaussian White Noise can improve the performance observed by [241]. Also, the spike count of the hidden population correlates with the input noise level because the same amount of spikes are generated during the

simulations above 35 dB, and below the spike count increases heavily. The forward architecture increases the spike count from 69 to 616 in the recurrent network by a factor of 3.2 up to 590. The increasing spike activity of the R&F encoding increases from 290 up to 2,300 spike events and therefore propagates the noise through the spike events. During a similar performance for low noise levels, the spike count of the encoding layer is constant. This indicates the direct influence of the filtering capabilities of the encoding method.

4.6 Discussion and Summary

This chapter uses the interference detection application to compare different encoding methods and architectures of traditional and spiking neurons. Thus, we define interference as an outlier from normality since it is a sparse event. Radar interference is the superposition of the reflections of objects with other high-frequency signals generated by another radar sensor or a deliberate attacker.

We propose two fundamentally different architectures for interference classification: the predictor and pattern classifier. The main difference is that the predictor can detect unseen outlier patterns, while the classifier can only detect trained outliers. The predictor represents the expected normality, and a deviation between the actual and predicted signal indicates an erroneous behavior. While the predictor architecture is semi-supervised, the classifier is supervised. The classifier detects specific patterns in the input signal to differentiate normality from interference, meaning the classifier can only detect known patterns. However, the spiking predictor approaches fail in learning the signal prediction with different encodings and decodings but performed very well in the classifier approach. The LSTM and SNN architectures work directly on the data stream without storing the input signal, showing that the local states of the network are enough to store the temporal features.

In the spiking classifier architectures, we observe that the SNN performs comparably to the CNN with preprocessing and outperforms the threshold-based detection. The R&F converts the temporal signal into a sparse spatio-temporal representation similar to the STFT. The conversion to a spatial spike train improves the training method compared to the current injection and TC encoding. However, the BPTT is less suited for long sequences, which need to be windowed to implement a continuous classifier. The continuous classifier is implemented in the spiking predictor but not in the classifier. We observe that spiking architectures need more training epochs than ANNs. Additionally, we identified that the CNN with the STFT preprocessing performs better than the pure CNN during limited network sizes.

Moreover, since it suppresses information, the R&F encoding affects the network's overall performance. Thus, we compare different spike and reset methods through extensive simulations. We observe that the phase-dependent spike mechanism does not help the classification in the example. This makes sense because the linear frequency changes identify interference, not the individual frequencies' phase. During the training of the SNN, we apply a spike regularization to minimize the spike count to increase efficiency.

There is a direct correlation between the classification accuracy and the spike count, but a slight regularization factor reduces the spike count significantly with a small accuracy drop. Also, we analyze how well we can prune the pre-trained architecture of SNNs and ANNs. We observe that the spiking architectures are more stable during pruning than CNNs, indicating that these networks are more suitable for safety applications.

Finally, we analyze the noise robustness of the different methods. Most methods are stable until an SNR of 35 dB except the R&F with graded spikes, which we explain by the propagation of the amplitude changes to deeper network layers. The pure CNN architecture performs an averaging of the input through the kernel, which improves the noise robustness. The STFT and the R&F can extract the frequency components in noisy conditions and filters the noise before the subsequent layers. The best-performing networks are less stable in noisy environments, indicating that the generalization could be better. The LSTM predictor also shows a high noise robustness with the adaptation of the outlier threshold.

With this application, we have shown that SNN can outperform current state-of-the-art networks by using sparse events and data-stream processing during resource constraints. The R&F encoding improves the classification immensely, and the SNNs are less susceptible to pruning. However, a trade-off exists between the spike counts and the network's accuracy. A careful selection of regularization improves efficiency significantly.

5 Phase Estimation with Spiking Neural Networks

In the previous chapter, we have shown that R&F neurons can convert the radar signals into a spike representation where the temporal changes of the frequency are encoded in the spike events. The interference detector proves the pattern classification capability of the encoded events. Within the radar, the frequency and amplitude are essential for the range estimation. The phase of the receiving signals carries information about the object's velocity or angular position. The relative reference of the phase under observation controls the receiving information, as we indicated in the state-of-the-art radar signal processing pipeline in Chapter 2.1.1.

We selected the phase detector as the application under review to investigate the phase information within a spike train. In such an application, the encoder translates the continuous input stream of a sinusoidal signal into a spike representation. An additional layer decides if the signal's phase is within predefined limits. A phase detector is crucial in applications like signal generation [243], ultrasonic detection [244], radar, and a lot more [245], [246]. The current state-of-the-art is the mixer-based phase detector [247], where the phase differences arise by mixing the two signals. The DC output of the mixer is proportional to the phase difference and can be converted with an ADC to a digital representation for further processing. Thus, the phase difference is available as an exact floating point value, but continuous processor interaction is necessary. In an FMCW radar, the transmitter's phase is highly important for different modulation schemes and changes during slight temperature variations. The individual channels receive different temperature effects due to their geographical location, and the continuous monitoring of the phase changes would increase the system's reliability. Currently, the phase is monitored offline after a significant temperature shift. Throughout the signal processing on the receiver side, the velocity and angle-of-arrival use the phase information. Phase non-linearity negatively affects the performance of the object classification. The target of this investigation is the conceptual analysis of SNN-based phase detection and not the application in the radar system because SNNs are currently not designed for the GHz range.

In the following sections, we investigate two current injection encoding schemes with LIF and R&F neurons while realizing a spiking phase detector and classifier. The weights between the two layers create the detectable phase boundaries utilized in the exact spike timing. Following the software investigation, we implemented the LIF phase detector and a dedicated analog circuit to analyze the efficiency. This work is part of our publication [42], except for investigating the R&F encoding.

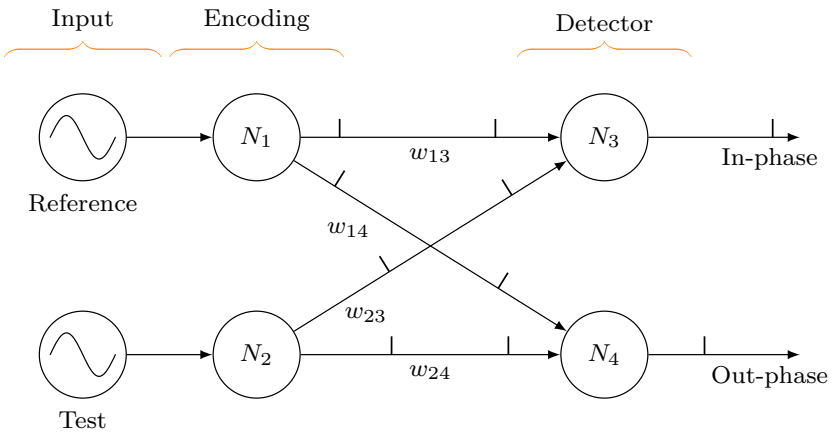

Figure 5.1: Phase detector architecture: System architecture of the spike-based phase detector. The encoding neurons convert the input signal into spike events. The detector neurons evaluate the temporal spike difference of the reference and test channel by the relation of the synaptic weights. The indices of the weights indicate the pre-and post-synaptic neuron. A spike event of neuron N_3 indicates that both signals are in the predefined phase limits and N_4 spikes when the signals are out of phase. A system failure is detectable by no output activity.

5.1 Concept and Experimental Setup

The following will describe the theoretical concept with information about the simulative experiments. Afterward, we demonstrate the hardware-specific properties of the proposed system and highlight the differences in the weight selection to the software.

5.1.1 Theoretical Concept

The spiking phase detector consists of two fundamental elements the encoding and decision layer. In Figure 5.1, we show the block diagram of the phase detector architecture. The encoding layer transforms the reference and test input signal with current injection into a spike train either with the inherent properties of the LIF or R&F neurons, and the detector neurons (LIF) decide whether the phase is within the expected limits. We refer to Chapter 2.2.3 for the mathematical details of the neuron types.

As mentioned, the LIF neuron fires when the neuron state exceeds the threshold potential, and the refractory period deactivates the neuron for a predefined time. Therefore, the threshold is selected such that spikes appear at the peak of the sinusoidal input signal. The refractory period suppresses a bursting behavior by preventing the leaking of historical information into the next period. We select the refractory by a quarter of the signal's period. Thus, the neuron continues to integrate during the negative input and reaches the threshold again during the positive part.

As another neuron under test, we use the R&F type, which acts like a frequency-selective filter element. When the neurons' resonate frequency matches the input, then the neuron oscillates and spikes depending on the selected spike mechanism. During the

interference detection in Chapter 4.5.3, we observed that the different spike mechanisms affect the classification performance and the number of spike events. The spike mechanism effects are negligible in the phase detector because only the relative spike timing of the two input neurons is essential. However, we use the phase-dependent spike mechanism where the neuron fires during the sign switch of the neuron's imaginary state since amplitude imbalances between the channels do not influence the output spikes. Afterward, we apply a reset to zero, and the oscillation begins to increase again. Less relevant is the choice of threshold because a threshold above the noise level ensures that only the resonant frequency stimulates the neuron and generates spike events. In LIF encoding, a change in amplitude due to channel differences can cause a shift in the pulse timing while the threshold remains the same, and the phase can no longer be detected. This problem cannot occur with the phase-dependent spike of the R&F. The refractory period is less important in the R&F encoding since the neuron needs longer to reach the threshold than in the LIF encoding. Therefore, we neglect the refractory period in the given application with R&F encoding.

Generally, both encoding methods generate spikes depending on the phase offset of N_1 and N_2, directly proportional to the spikes' time offset Δt. Therefore, the phase φ can be calculated for a specific frequency f as follows:

$$\varphi^\circ = 360^\circ \, f \, \Delta t. \tag{5.1}$$

This correlation demonstrates that the spikes of the encoding layer must carry the information of the time shift between the inputs for phase detection. The input layer supports sinusoidal, pulse, or spike-based input signals.

The second layer decides whether the phase exceeds the limits by using two neurons that indicate in-phase (N_3) or out-phase (N_4) detection. Thus, the LIF neurons act like coincidence detectors that are biologically explainable [248]. It is possible to use a single neuron. However, from a functional safety perspective, such a two-neuron system provides a reliability check to whether the encoding and synaptic connections are working. However, detecting the spike activity of the encoding layer would also indicate the encoder's functionality but nothing about the synaptic connection. If zero spikes appear at the output layer, then it means that the functionality is disturbed, and the system should be checked. The connections between the neurons are denoted depending on the pre-and post-synaptic neuron indices, where all weights expect w_{14} are excitatory.

The temporal correlation of the spike timing, the weights, and the neuron leak decides whether the spike timing difference is within predefined limits. For example, the spike from N_1 appears at t_1, and from N_2 appears at t_2 with the temporal shift of Δt. The membrane potential of neuron N_3 increases at t_1 and leaks until the second spike increases the membrane potential again. Due to the exponential leakage defined by the time constant τ_m, the weights w_{13}, w_{23}, and the relation between the spike events define the maximum membrane potential of N_3. We can express the decaying effect of the membrane potential by the following simplification:

$$V(t) = w \, e^{-t/\tau}, \tag{5.2}$$

Table 5.1: Phase detector neuron parameters: Default parameters for the LIF neuron applied in the encoding and detection layer. The second table contains the default parameters of the R&F without an exact resonate frequency.

LIF	
Parameter	**Value**
V_{th}	1.0 mV
V_{reset}	0.0 mV
C_{m}	50.0 pF
τ_{m}	7.0 ms
t_{ref}	10.0 ms
τ_{syn}	2.0 ms

R&F	
Parameter	**Value**
V_{th}	2.0 mV
V_{reset}	0.0 mV
b	1
f	various

where w is the weight of the first spike event, and we assume the same weights for both connections. The simplification neglects the membrane capacity and the exponential post-synaptic current model. However, it demonstrates the close correlation of the weights, the membrane time constant, and the threshold. When the membrane potential increases with the second spike and exceeds the threshold, the neuron fires; otherwise, it leaks to the resting potential. In the case of an action potential, a refractory period keeps the neuron inactive to ensure a single spike event during one period of the sinusoidal signal.

The out-phase detection uses the same properties of the LIF neuron. The first event uses the inhibitory synapse w_{14} to invert the leakage from a negative to the resting potential, and the second spike uses a strong excitatory connection. Therefore, the neuron N_4 spikes when the second incoming spike is strong enough to raise the inhibited membrane potential above the threshold. If the second spike occurs before the first spike, the neuron will always fire due to the strong excitation, but for an in-phase signal, the neuron N_3 also fires. Due to the exponential post-synaptic current, there is a delayed charging which means a later inhibitory input influences the membrane potential and suppresses a spike event. However, there are three solutions to the problem: introducing another neuron with inverted weights to catch the out-phase for the earlier test signal, detecting this problem, and switching the weights to N_4. Another solution is to use only the in-phase detection and verify the system's functionality with the spikes of N_4 during out-phase scenarios. The out-phase weights are generally more difficult to select due to the imbalance between the two synapses. Figure 5.1 shows two examples of in-phase and out-phase detection within two separate windows. This is possible because we ensure that all neurons reach the resting potential in less than a period.

5.1.2 Simulation Setup

For the conceptual work of the phase detector, we use the nest-simulator [249]. Since there is currently no implementation of the R&F neuron, we circumvent this problem by

encoding the signal with the PyTorch model and propagating the spike events directly to the synapses w_{13}, w_{14}, w_{23}, and w_{24}. Thus, our pipeline comprises a signal generator that generates a test and reference signal depending on the evaluation with additional noise, signals, or synthetic disturbances. In the case of the LIF encoding, the generated signals are directly connected with "iaf_psc_exp" neurons with the parameters from Table 5.1. For the R&F encoding, we forward the output of our PyTorch model to "spikegenerators" that replace the LIF encoding. In Table 5.1, we indicate the frequency parameter of the R&F neuron, which depends on the experiment and can vary slightly from the actual signal frequency. All other components are the same for both encoding methods, like the synaptic connection, modeled as "static_synapse" with an adjustable delay fixed in the current experiments to 1 ms. For a good visualization, we relax the phase limits to $\pm 12°$ and set the weights for the following experiments to $w_{13} = 21.15$, $w_{23} = 21.15$, $w_{14} = -20.5$, and $w_{24} = 59.85$.

Important to note is here that these parameters are scalable to different requirements. Further individual changes to the default settings are explained in the corresponding experiment.

5.1.3 Hardware Implementation

The pure evaluation of the software-based concept estimates the usability, speed, costs, and power consumption. Therefore, we have realized the previous concept of phase detection in analog hardware in a 130 nm BiCMOS process. Currently, the research mainly focuses on analog LIF neurons instead of R&F neurons. Due to the limited development time and the availability of LIF circuits from previous work [49], we decided to implement the LIF phase detector as a proof of concept. In the following chapter, we limit the explanation of the hardware implementation to an understanding of the later evaluations and the differences in the software concept. We refer to our publication for more details [42]. The circuit consists of two fundamental building blocks the LIF neuron and the synaptic connection. The block diagram in Figure 5.1 demonstrates the alignment of the individual modules. For interested readers we provide a sophisticated block diagram in our publication [42] in Figure 4.

Different dynamics exist for the LIF neuron to model different biological plausible dynamics. We modified the circuit of [250] to optimize the area by removing the spike adaptation feature. The original circuit uses a spike frequency adaptation with an optional threshold and leak. While the spike frequency adaptation [251] has a high effect in rate-encoded systems, it is ineffective with single spike events in the phase detector application. The circuit can be divided into three components: the integrator with a leak, the spiking mechanism, and the refractory mechanism. Instead of the original refractory mechanism, we have added a transistor to decouple the refractory period from the spike generation to ensure invariant output spikes for the subsequent synaptic modules.

Additionally, the refractory mechanism resets the membrane potential after each spike event to prevent spike bursts. Due to technical reasons, the capacitor's minimum dimensions are 20 fF, allowing a maximum spike frequency of 70 MHz. The cell size is

19.9 μm x 13.2 μm. The individual neuron has three freedom parameters to adjust the functionality and response.

Each synaptic connection transports a spike event from a pre to a post-synaptic neuron and applies a weighting function. In simulations, adding the weights to the input current during a binary event is sufficient. Therefore, the analog implementation has the following requirements: 1) a pre-synaptic spike triggers a synaptic output current with 2) a variable amplitude and 3) an exponential decay by the synaptic time constant. The excitation of the post-synaptic neuron depends on the weight that emerges from the synaptic current's amplitude and period. As a synaptic model, we use the differential pair integrator from [252] that satisfies our requirements. Two voltage-controlled transistors affect the synaptic weight and time constant, supporting global homeostatic plasticity mechanisms not used in the phase detector. We limit the weight adjustment to the time constants to reduce the parameter space. The synapse does not support an adjustable time delay, which the phase detector does not require. Here the synapse consists of three parameters for the highest flexibility but shares two between all synaptic connections.

A key difference to the software implementation is the polarity of the synaptic weight by acting as an excitatory or inhibitory connection. In software, the sign of the weight sets the synapse direction, but in analog hardware, the sign describes the direction of the current flow. For higher flexibility, the same differential pair integrator synapse can lead to excitation or inhibition depending on whether the synapse is connected to the current input or the neuron's leakage path. An inhibitory spike event increases the discharge of the membrane potential via the leakage path and prevents the excitation due to incoming excitatory events during the duration of the leakage. However, it is unlikely but possible that strong excitatory spikes lead to a spike generation.

The phase detector comprises four neurons and four synapses, leading to 24 adjustable parameters. Due to sharing the threshold, synaptic output current, leakage, and time constants, we can reduce the parameter count to 8. However, this reduces flexibility, and the synaptic weights are entirely defined by the time constant or the current flow time. In our publication [42], we show a photography of the fabricated chip.

5.2 Evaluation

We have separated the evaluation into the encoding layer and the phase detector system. In the first part, we focus only on the differences between the two encoding methods, followed by the analysis of the proof of the concept and measurements of the spiking phase detector implemented in analog hardware.

5.2.1 Encoding Layer

The phase detector consists of two layers, each with a different task. In the following experiment, we focus on the encoding layer and highlight the differences between the current injection method of the LIF and R&F neurons by simulations. We perform each experiment for 1 second and select a representable part of the simulation. In the following, we disregard the unit of the amplitude since this is just a scaling factor.

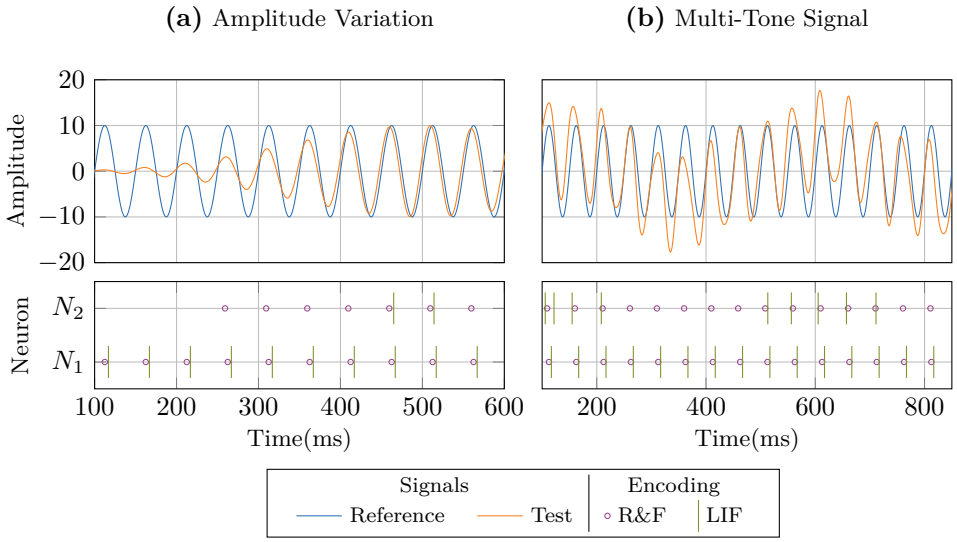

Figure 5.2: Encoding differences by signal variations: Analogy of R&F and LIF encoding for input signal variations like (a) amplitude changes and (b) the superposition of multiple tones. The upper plot shows the input signals from the reference and test source, and the lower plot demonstrates the simulated spike events of the encoding layer.

Amplitude Variation

In the first experiment, we analyze the encoding response based on the test signal's amplitude. For a clear visualization, we use a 20 Hz signal and apply a phase shift of 20° to the test signal. We generate an amplitude change of the test signal by a Kaiser window with $\beta = 10$, as demonstrated in the first plot of Figure 5.2 (a).

The LIF encoding shows lower flexibility in terms of amplitude changes. In Figure 5.2 (a), the LIF neuron only fires during the highest amplitudes for the given input. In contrast, the R&F neuron generates spikes for a lower amplitude level. A further increase in the amplitude of a single channel introduces a temporal shift of the spike timing, reducing the phase resolution of the LIF encoding. While the amplitude is above the expected noise level, the R&F neurons generate spikes conditional to the phase. In this example, the spikes of N_2 appear earlier than N_1, and an additional amplitude increase does not influence the spike timing. Therefore, the threshold selection depends only on the expected noise level, unlike in the LIF encoding on the actual amplitude.

The expected spike distance of the 20 Hz signal is 50 ms due to Equation 5.1, and the average spike distance of the R&F encoder is close with 50.08 ms and a standard deviation of 0.063 ms. Therefore, the maximum phase error of the R&F spike events is 0.45°. In the current example, the LIF generates only two spike events with a distance of 49.2 ms. Important to note is that this metric is the phase change across spike events of a single channel and not the relative phase between the channels.

Since the threshold of the LIF only depends on the signal's amplitude, a change in amplitude or imbalance between the reference and test channel introduces a jittering of

the encoded spike. Such spike time variation does correspond to a low phase resolution. The frequency-selectivity of the R&F neurons increases the reliability of the output events while changing the amplitude.

Frequency Selectivity

During the second experiment, we investigate the benefit of the frequency-selective properties of the R&F neurons. Therefore, we use the reference and test signal with a frequency of 20 Hz and a phase difference of 20°. We add a signal with a 5 Hz frequency, an amplitude level of 5, and a phase shift of 10°. The superposition of both signals is visualized in Figure 5.2 (b).

Here, we use a very simple signal superposition demonstrating the differences between the two encoding schemes. The LIF depends only on the relative amplitude of the input signal, whereas the R&F is frequency and amplitude selective. Since the input shifts into the negative domain, the LIF does not emit spikes between 200 and 500 ms. Also, at around 120 ms, the LIF generates an additional unwanted spike because the first spike was very early, and the refractory mechanism could not protect a second spike. On the other hand, the R&F is unaffected by the additional sinusoidal signal. The average spike timing of neuron N_2 is 44.17 ± 11.63 ms and 50.07 ± 0.64 ms for the LIF and R&F, respectively. For the LIF, we neglect the long temporal difference, as visible in Figure 5.2 (b). The R&F achieves the expected temporal difference defined by the signal frequency. However, the phase of the test signal changes slightly due to the superposition of the two signals with different phases and frequencies. The R&F encoder has the advantage of frequency selectivity, providing a more comprehensive application range than the LIF.

Noise Robustness

Noise affects the encoding performance during the actual deployment of the LIF and R&F current injection encoding techniques. We implement the same signal properties as in the previous experiments but with an additional Additive white Gaussian noise (AWGN) source and an SNR of -10 dB, demonstrating a scenario where the signal power is below the noise level.

The selected results of the experiment are visualized in Figure 5.3. As expected, the LIF neuron emits random spike events around the peaks of the sinusoidal signal. The average distance between the events of N_2 is 59.5 ms with a standard deviation of 26.24 ms, which can also be observed in the lower plot of Figure 5.3. We assume that the spike distance of the reference neuron N_1 is linear. Therefore, we can see that at about 365 ms, the spike event is more or less simultaneous with a distance of 1 ms to N_1. The spike of the following event appears 8 ms earlier, at about 408 ms. The difference of 7 ms is proportional to a phase difference of 50°.

The average spike distance of the R&F encoder is 49.97 ms, very close to the expectations and with a standard deviation of 1.9 ms considering the noisy circumstances. The pure visualization of the spike timing does not show the jittering of the R&F output events. However, for the same reference point as in the previous example, the spikes

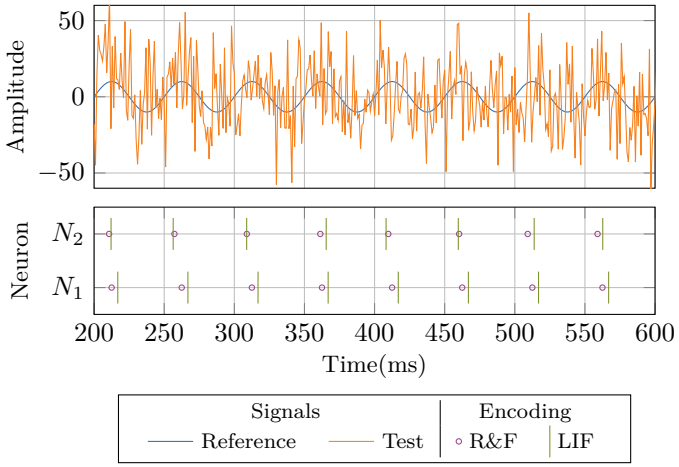

Figure 5.3: Encoder noise sensitivity: Differences between the sensitivity of the encoders to white Gaussian noise with an SNR of −10 dB on the test signal. The upper part shows the reference and test signal, and the lower plot the spike response of the LIF and R&F neurons. The R&F spike timing is less noise-sensitive than the LIF since frequency-selective properties filter the wideband noise.

appear 1.2 ms earlier at 361.3; for the following example, the reference event appears 2.7 ms later. The difference of 1.5 ms corresponds to a phase shift of 10.8°.

Thus, the spike timings of the LIF and R&F encoders are affected by wideband noise. However, the filtering properties make the R&F more stable with a lower phase error. Also, frequency spurs, close to the resonate frequency of the R&F encoder, highly affect the spike timing quality.

5.2.2 Phase Detection

We decided to use the pure LIF network to prove the concept of the phase detector. The encoding and decision layer use the same neuron type to simplify the analog circuit design and reuse existing LIF implementations. Therefore, we first demonstrate the simulation of the concept in the following section. Afterward, we focus on hardware measurements and specific properties of the implementation. These results are previously published in [42].

Concept Simulation

We use a neuromorphic simulator to investigate the in-phase and out-phase detection with different temporal shifts for the functional verification of the phase detector architecture. Also, we focus on the corner cases when the phase difference is close to the limits. Therefore, we use a test and reference sinusoidal signal with phase settings of ±20° and ±5°. Figure 5.4 demonstrates four possible cases with the different phase settings. We differentiate between the leading signals and the expectation of in- or out-phase detection. Since we have already analyzed the encoding layer in the previous section, we will not

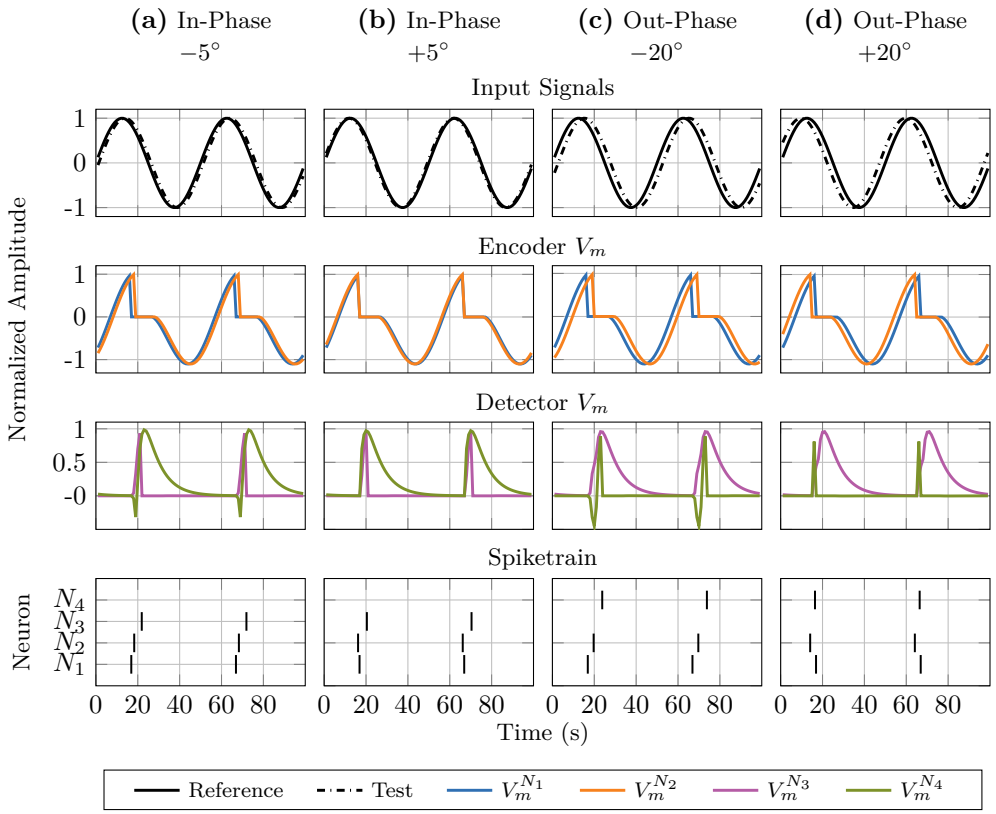

Figure 5.4: Simulation results of the phase detector: Simulative response of the neurons inside the phase detector with in-phase in (a), (b) and out-phase (c), (d) examples. We increased the accepted phase difference for better visualization. The last row demonstrates the spike events of each neuron, where N_3 indicates in-phase and N_4 out-phase detections. Modified version from our publication [42]. It is recommended to view this figure in color.

go into more detail. All simulations show a clear spike response with a followed dead time due to the refractory period. This suppression mechanism prevents the generation of more than one spike per period.

In Figure 5.4 (a), the test signal reaches the maximum later, shown by the earlier spike event of N_1 than N_2. The temporal delay between the two sinusoidal inputs is within the selected limits, indicated by the spike event of N_3. It differs from (c), where the phase is outside the limits, triggering an event by neuron N_4. The concept of the synaptic membrane potentials of the detector neurons shows the interplay between the synaptic weights, spike timings, and membrane time constants. Due to the close spike events, $V_m^{N_3}$ increases fast until the threshold and is reset. The simulation can be misleading because the simulator immediately resets the potential and does not store the maximum membrane potential of the neuron, explaining the immediate drop without reaching the threshold voltage. In (a), the membrane potential of N_4 decreases enough to prevent an output spike. We see that we are close to the preselected phase limit from the amplitude

level. Therefore, in Figure 5.4 (c), the inhibitory weight is small, and the delay of the excitatory spike leads to a decay of the membrane potential, leading neuron N_4 to fire. At the same time, the temporal distance of the input spikes is too high to generate an output event at neuron N_3.

Columns (b) and (d) in Figure 5.4 show the case when the test signal is before the reference signal. The in-phase detection does not change by the order of the incoming spikes since both synaptic weights are the same, which explains that we can observe a similar response of the membrane potential of the detector neurons in (a) and (b). Also, the membrane potential of N_4 demonstrates the reason for a wise selection of the time constant to prevent information leakage into the next event and drift of the detected phase. In contrast, we see in (d) a proper detection of the out-phase without the inhibitory effect at N_4. N_3 indicates an out-phase scenario by zero spike events, while the neuron N_4 fires. The synapse w_{24} is excitatory with a high weight to overshoot the inhibitory effect of synapse w_{14}. Because the inhibition is too late to decrease the excitatory stimuli, the membrane potential of neuron N_4 exceeds the threshold and fires. In (d), the neuron N_4 fires, and the inhibitory spike arrives during the refractory period. Thus, out-phase detection is vaguely defined for an earlier test signal, a limitation of the current concept. This vague out-phase detection can lead to a simultaneous firing of both output neurons while the phase is still in the expected range. For example, with a phase shift of the test signal by $+10°$, both neurons fire simultaneously. However, the indication of neuron N_4 is only a robustness check of the system, and in such a case, a switch of the weights to N_4 is sufficient.

A failure such as the signal amplitude changes, problems in the encoding, or with the weights leads to an anomalous activity of the output neurons. Since we have two output neurons that fire for in-phase or out-phase, we can detect failures affecting both neurons' spike activity. An example is that the loss of a complete input channel will lead to zero spike events.

Hardware Measurements

We use comparable scenarios to the software simulations for the fabricated test chip to validate the functionality. We extend the analysis by a behavioral experiment of the circuit during a channel failure. Nevertheless, there is also an analysis of a clock-based input shown in [42].

For the functional validation of the circuit, we use an example where the reference and test signals have a signal frequency of 125 kHz with an amplitude of 1 V. Because we can only use positive voltages, the signal oscillates between 0 and 1 V. The threshold and weights are adjusted to set a phase boundary of $\pm 7°$ for better visualization. We report the response to a shift of $5°$ and $20°$ phase shift as shown in Figure 5.5. Since the area of the pads exceeds the circuit itself, we are limited by the number of pads to control the circuit. Thus, we have only two pads to read the output of neurons N_3 and N_4.

Figure 5.5 (a) demonstrates an in-phase example where neuron N_4 is inactive and N_3 fires. In contrast to the software simulation, the spike is not anymore a binary event; instead, we see a discharge over time. This widening of the spikes appears due to parasitic

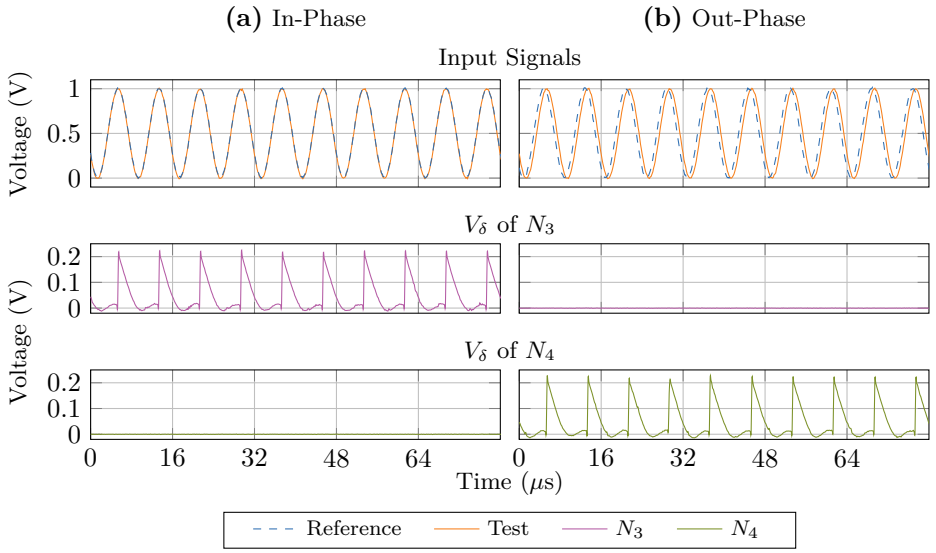

Figure 5.5: Hardware measurements: Measurements of the fabricated spiking phase detector hardware. (a) is the example of an in-phase detection, while (b) is the out-phase scenario with a phase shift of 20°. The spike response of the neurons demonstrates the detection of the phase shift. Modified from the publication [42].

capacitance in the test chip and measurement setup, which we verified by simulations. Still, a spike can be differentiated from no spike very easily. The second example (Figure 5.5 (b)) shows an out-phase scenario with a slight temporal shift of the test signal. This shift is far outside the expected phase limits; therefore, N_3 is inactive and N_4 active. The output spike shape of the in-phase and out-phase detection of N_3 and N_4 is comparable. Also, we could observe the same effect as in the software simulation with the simultaneous firing of both neurons. As expected, both neurons fire when the reference spike appears later than the test spike close to the positive phase limit.

The circuit supports a maximum signal frequency of 70 MHz to generate a single spike per period. However, in [42], we demonstrated that the phase detector could also handle frequencies up to 200 MHz by using the refractory period of the encoding layer. The refractory period prevents that each period generates a spike event, and the decision layer receives input spikes with a lower frequency and can discharge between the events. Sudden phase changes during the refractory period are not detectable anymore.

We measure the circuit's response to a channel loss to demonstrate the built-in monitoring. Hence, the reference and test signal use the same properties as the previous experiment with a phase shift of 5°. At approximately 42 μs, the test signal abruptly ends and continues with a small offset.

Figure 5.6 shows the immediate end of the test signal's oscillation. Before the 42 μs, the circuit functions as expected because the signals are in phase, and we receive output spikes of N_3 and N_4 is constantly close to zero. With the channel loss due to the oscillation

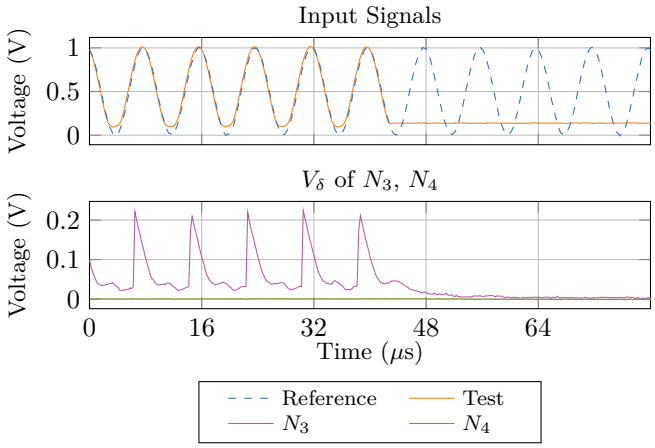

Figure 5.6: Phase detector channel loss: Hardware measurement demonstrating the built-in monitoring feature by an ending sinusoidal test signal. Neuron N_3 detects that reference and test signals are in phase until 42 μs. Afterward, no spikes are generated. Modified from the publication [42].

ending immediately, no further output events appear, and the output decays back to zero.

Because the encoding does not generate spike events when the signal's amplitude is below the threshold, and the binary events neglect the amplitude information, we can estimate the circuit function by the output activity. Therefore, the output will be zero when a single channel does not generate any spike events. The output shows a short increase and decrease of the potential between each spike event. With the immediate stop of the test signal, the output voltage increases and then decays to its resting potential. In the case of an out-phase detection, the spikes of neuron N_4 demonstrates the function of the circuit.

However, the identification of a failure is only limited to zero spike events. If more spikes, as expected, appear in the decision layer, it is not possible to differentiate between such a simple spike or no spike comparison. In such a case, observing the temporal changes of the spike response is necessary.

The phase detector consumes 840 μW calculated by the current consumption of the voltage source, measured during the 200 MHz signal with a minimum resolution. The active area of the circuit is 61.38 μm x 38.4 μm without the test chip pads.

5.3 Discussion and Summary

In this investigation, we answer the research question 2, which asks whether SNNs can differentiate the frequency and phase by using the phase detector application. A phase detector compares the relative phase between a test and a reference signal and

is important in various systems. The proposed phase detector combines the detection and evaluation without necessary processor interaction. Ideally, the output spikes could adjust the system phase or be converted to a digital representation.

In contrast to SNNs, traditional ANNs suffer from temporal properties and, therefore, cannot differentiate the phase differences of an input stream without additional memories. Such memories would be necessary to convert the temporal dimension to a spatial dimension and use a dense or convolution input layer by storing at least a period of the input signals. Thus, the advantage of the proposed SNN is the continuous evaluation of the phase and the sparse event activity.

In this thesis, we have extended the previous publication [42] with the comparison between LIF and R&F encoding. While the LIF neuron is widely used, the inherent properties of the R&F provide some relevant aspects. The frequency selectivity of the R&F neurons enables the encoding of multi-tone signals and a higher resistance to white Gaussian noise. The encoder replacement does not affect the second layer as long as the spike time difference between the channels is proportional to the relative phase. The R&F neurons can also be implemented in the detector layer and detect the frequency of the encoded spike events across the channels. Therefore, only two events stimulate the neuron, where the weights, damping constant, and threshold defines the phase limitations. The in-phase neuron function is comparable to the LIF, but the out-phase neuron has one more dimension. However, we expect that the phase detector is not limited to these types of neurons, and with architectural changes, other neuron types can be used.

We implement the LIF phase detector to validate the concept in analog hardware. In contrast to the simulation, the weights correspond to the current flow's duration instead of the current amplitude. Since the capacity of the neurons and synaptic connections should exceed the parasitic effects, the maximum frequency of the circuit is 70 MHz. Thus, we prove that artificial SNNs could work with higher frequencies than their biological counterpart with 500 Hz [253]. Also, the hardware implementation works continuously while the software simulation uses a time step size which introduces a spike timing error.

The two output neurons generate spikes depending on whether the phase is within the predefined limits. This two-neuron system provides the advantage that problems with the input, encoding, or synaptic connections lead to zero spike events, as demonstrated by the channel loss example. The out-phase is vague for the case when the test signal is before the reference signal because the inhibitory effect of the synaptic weight from the reference appears after the test signal spike. Therefore, the earlier test signal leads to an output event. Thus, there is a region where both neurons fire, meaning that the signals are within the phase limits. An additional neuron could indicate the positive phase change, but the differentiation between the scenarios depends only on the in-phase detector neuron. Therefore, switching the weights in such a case is better or focusing mainly on the in-phase detection and using the out-phase neuron only as a functional check.

The study by [254] demonstrates a spiking phase detector that is integrated into a neuromorphic phase-locked loop, enabling the conversion of temporal coding into rate coding. This phase detector employs rate coding to represent the phase difference. However, this system employs a population of neurons with varying synaptic delays,

resulting in increased area and power consumption. In contrast, our approach focuses on the classification of in-phase signals. By this, the closed loop can adjust until neuron N_3 fires with an integrated encoder that directly connects the analog signal to the phase detector. Notably, while [254] only provides examples within the 35 Hz range, our proposed phase detector supports a maximum frequency of 200 MHz, enabling phase decision without requiring processor interaction.

6 Spiking Radar Angle-of-Arrival Estimation

Radar applications like collision detection are only helpful if the signal processing algorithm knows if the object is in front or aside from the ego vehicle. Thus, many applications, like radar, Ultrasonic, and even stereo-cameras, must differentiate the angle of different objects within the field of view.

In the following, we propose a spiking angle estimation method that emits temporal spike events instead of single floating point values. The research is motivated by the ongoing improvements in angle estimation with deep learning methods and the inspiration of the bats' and dolphins' excellent and efficient echolocation abilities. In [72], the authors compare different deep learning implementations and observed higher performance with increased computational complexity. Thus, we present the first step towards a spiking angle estimation method inspired by improved energy efficiency compared to traditional NNs. Thus, we define the following assumption while developing and analyzing an end-to-end spiking angle-of-arrival estimator: 1) The modulation scheme can be neglected because we only use a single transmit channel and multiple receiving antennas. 2) The range-angle map can be estimated on a single chirp to neglect phase shift by velocity effects. The velocity estimation introduces the necessity of long-term memory in the stream processing pipeline.

During our research, we realized that there are three possible solutions for multi-target angle estimation. The first idea was the usage of a tracking algorithm to interpolate between measurements of the target position. However, in some scenarios, two objects appear simultaneous and are not trackable. The second idea was to train a NN for angle detection, and the third was the usage of a second R&F layer. We trained a traditional CNN on the R&F output by storing the time steps, and the network learned the angles but was not very good at generalizing them to more angles. Still, we had the goal of accomplishing a complete spiking solution. Therefore, we also trained multiple networks of LIF neurons with surrogate gradient and an evolutionary algorithm with a hyperparameter search. All tested architectures failed to solve the task. The last solution used a second R&F layer with complex synaptic connections that we explain and investigate in the following chapter. The network demonstrates the use of the frequency, amplitude, and phase information of the radar signal components by reusing the previously described characteristics of the R&F neuron.

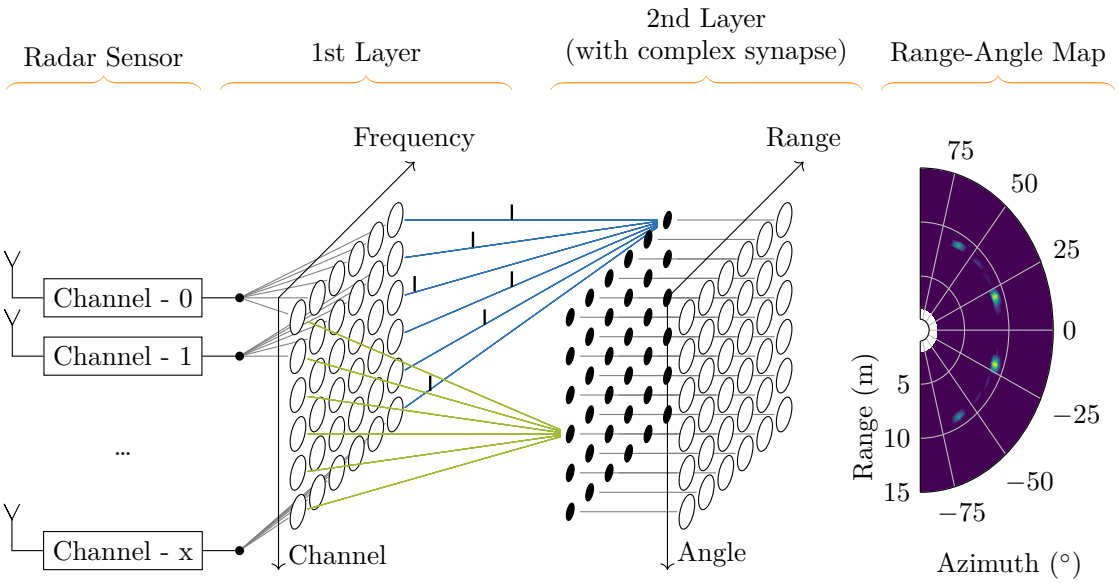

Figure 6.1: Architecture of the spiking angle estimator: The end-to-end architecture for angle estimation consists of two R&F populations. The first R&F layer converts the signal of the individual receive channels to spatio-temporal events. The output events are weighted by complex synapses and forwarded to the second layer R&F neurons. For clarity, the full connectivity between the first layer and the complex synapses is reduced to two example mappings from the channel to the angle dimension. The range-angle map shows a simulated scenario's normalized accumulated spike events with 32 channels and two targets with constant radar-cross-section at a $\approx$ 9-meter distance with an angle of $\pm 20°$.

6.1 Method

We propose a fully spiking architecture with R&F neurons for the angle estimation with a spiking network. However, also the spiking networks underlie the physical properties of the receive channel while we have to mitigate the introduced differences between the channels. The channel variations occur because of fabrication tolerances and temperature differences between the channels through their geographical position. So, we first describe the proposed two-layer R&F architecture with the complex weights, then explain a possible calibration method.

6.1.1 Network Architecture

The proposed network architecture, shown in Figure 6.1, uses the previously analyzed R&F neurons where the first layer performs two tasks: 1) the separation into the different frequency components and 2) the conversion of the radar signal to spatio-temporal spikes where the spike timing correlates with the phase and a unit amplitude. For each individual receive channel, a population with linear scaled resonant frequencies is sufficient and equivalent to the Fourier transformation as investigated in Chapter 3.4.4.

The angle of a single target at a distance r is proportional to the phase difference between two consecutive channels, corresponding to a temporal difference between spike events of two identical R&F neurons of different channels. The temporal difference is defined by:

$$\Delta t = \frac{d_n}{c} \cos(\theta), \tag{6.1}$$

where θ is the angle of the target, c is the speed of light, and unitless d_n distance factor between two antennas where the actual distance depends on the wavelength and is defined by $d_n \cdot \lambda$. The evaluation contains more details about the angle estimation of a single target than the traditional Fourier transformation. However, in a multi-target scenario with more than one target, the received signals are a superposition of the individual echoes. A frequency analysis of the phase rotation across the channels separates the individual targets. An FFT across the channel dimension in traditional signal processing extracts the azimuth angle. More information about traditional signal processing is provided in Chapter 2.1.2. Before using the radar cube to process the angle, it is necessary to reorganize the data based on the physical location of the antennas.

In the end-to-end spiking angle estimator, we use a second layer of R&F neurons with complex input, which means the neuron receives an imaginary $\Im$ and real $\Re$ input. The preceding complex synapse implements a matrix multiplication of the transposed time-dependent pre-synaptic spike matrix $\boldsymbol{X}$ and the complex weights $\underline{\boldsymbol{W}}$ of the dimensions $N_u \,\mathrm{x}\, N_c$. N_c represents the number of physical receive antennas, and N_u is the number of expected angles. Hence, the input to the post-synaptic neuron is defined by:

$$\underline{\boldsymbol{Y}}_{i,u}[t] = \sum_{c}^{C} X^T_{i,c}[t] \, \underline{\boldsymbol{W}}_{c,u} \,. \tag{6.2}$$

We precalculate the weights based on the known distance factor $d_c = d \cdot i$ of antenna c to the reference. Notably, the complex weights represent the phase rotation of an imaginary target with a specific geographical angle regarding the receive antenna. Therefore, the weight between antenna c and angle θ_u is defined by:

$$\underline{\boldsymbol{W}}_{c,u} = e^{-j2\pi d_c \sin(\theta_u)} \,. \tag{6.3}$$

Depending on the antenna and expected angle, the complex weights have a unit magnitude and a phase. Depending on the phase rotation of the complex weight and the oscillation of the R&F neuron at a specific spike time, the phase rotation either strengthens or diminishes the oscillation. The most significant magnitude change occurs if the weight vector is orthogonal to the tangent of the oscillation. This means that it can either increase or decrease depending on the interaction of the sign of the weight and the oscillation at the spike event, as depicted in Figure 6.2. A slight modification of the timing reduces the magnitude change during the complex weights. However, the complex weights solve two problems: 1) the mapping of the physical antenna position and 2) the temporal order of the spike events because the synaptic weights consider both. Without the complex weights, the temporal difference between two spikes decides the

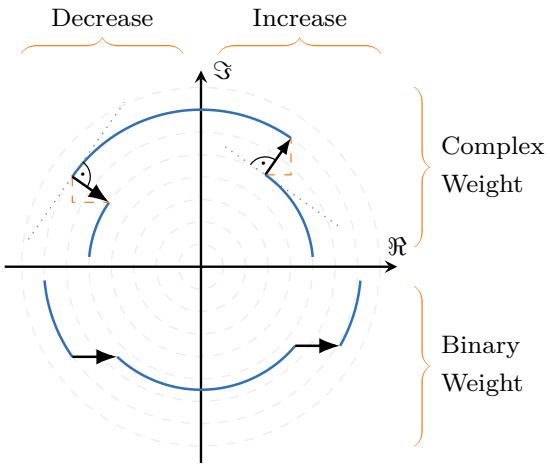

Figure 6.2: Difference between binary and complex weights: Theoretical differences of the complex and binary weights for a decrease and increasing magnitude. The neuron oscillation (counterclockwise) is constant for the different weights, and the weights have a unit magnitude. If the complex weight of the event is orthogonal to the oscillation, it receives the most significant magnitude change without affecting the phase. The binary input modifies only the real component of the neuron state and therefore introduces a magnitude and phase shift.

change of magnitude. For example, the antenna position and object location rotate the phase by 180°, leading to a magnitude decrease rather than an increase. Also, Figure 6.2 demonstrates the effect of a binary input on the real component of an oscillating neuron. The neuron state receives a magnitude and phase change if the input event is not precisely at $\Im = 0$. The sign and timing of the input decide if the magnitude increases or decreases for both weight settings.

Resonant frequencies of the second layer of R&F neurons are identical to those of the first layer. The differences between the angles are only detected by the complex weights' stimuli, which reduces the design's complexity. In order to further reduce the number of parameters, all complex weights can be scaled to match the necessary threshold between the layers (here by 10^2). As previously, we map the radar input directly to the first R&F population without modeling any synaptic connections. Also, the calculation of the complex weights is independent of the resonant frequency (see Equation 6.3) and, thus, can be shared across the range dimension. This means the blue and green connections in Figure 6.1 have an identical weight vector.

Consequently, the architecture consists of two layers with a total of $2C \cdot (N + U)$ states, where C is the number of channels, N is the number of range frequencies, and U is the angle count of the output. Optionally, the output can be interpolated to more angles than existing channels, and without calibration, the complex weights are symmetrical

across the angle dimension. The number of parameters can be calculated by:

$$N_{\text{total}} = N_{th}^{(1)} + N_b^{(1)} + N_f^{(1)} + 2CU + \underbrace{N_{th}^{(2)} + N_b^{(2)} + N_f^{(2)}}_{\text{without sharing}}, \tag{6.4}$$

which includes each layer's threshold, damping, resonate frequencies, and the complex synapse. In order to reduce the memory footprint in digital implementations, we can apply parameter sharing between the R&F layers and remove the parameter count of the second layer.

In the example architecture of Figure 6.1, the output represents a range-angle map with the normalized accumulated spike events across the time of the second R&F layer. The output can be directly used for applications like object classification and target tracking instead of applying the optional decoding.

6.1.2 Calibration

The high-frequency components of the radar MMIC are sensitive to fabrication tolerances that affect the balance of the different receive channels [255]. Also, the geographical difference between the channels results in a different temperature behavior which introduces a varying phase shift between the channels [256], [257]. In order to enhance the performance of object detection, it is necessary to compensate for the imbalance between the receivers. Several methods apply to the transmit channel for the usage of multiple transmitters that we leave unconsidered. The mitigation of the phase and amplitude imbalance is mostly considered in the design of the signal processing pipeline. One possible calibration setup calculates the phase and amplitude differences during a known measurement where the object is at zero degrees with a known distance [258]. The calibration vector is extracted from the deviation of the magnitude and phase of the range data across the channels and has to be recalculated during temperature changes.

Similarly, we can implement a calibration of the spiking angle estimation method. The amplitude differences only affect the range calculation and thus can be mitigated by modifying the threshold values of the first R&F layers. However, the R&F neurons are less sensitive to small amplitude changes because the focus is on the phase and existence of the frequency. When big amplitude differences exist, the time-to-first spike of the individual frequencies across the channels provides information about the increase and decrease of the threshold. This method only approximates the amplitude imbalance because the exact imbalance does not improve the system's performance.

More important is the phase imbalance that mainly affects the angle estimation and, therefore, the second layer of R&F neurons. The phase imbalance is an unwanted temporal delay between the channels. It leads to two problems: 1) the shift of the angle and 2) the blurring of the angle estimation across the adjacent angles. There are different methods to compensate for the phase imbalance, for example: introducing a synaptic delay from the input to the first R&F layer. The same delay can be spread across the synaptic connections between the layers but the complexity scales with the number of channels C, and for the between-layer delay, also it scales by NC. A better-suited method

reuses the complex synaptic weights by multiplication with a calibration vector of size C. We obtain the calibration vector similar to traditional methods by calculating the magnitude and phase deviation of the different channels after the range estimation. Here is again the scenario known with a single object at a known distance and an angle of zero. Therefore, the calibration vector introduces only a phase shift and neglects, for now, the magnitude.

6.2 Experiments

The following section further analyzes the proposed angle estimation with R&F neurons. All experiments use the same architecture with different settings depending on the showcase and the distances of interest. We start with the information analysis of the first layer only with an extension of a trainable network to demonstrate that the spike train carries information about the angle. Additionally, we have verified that a traditional MLP and CNN can estimate the angle based on the binary output of the R&F encoding layer. During our research, the traditional networks can only detect the event correlations while receiving all events simultaneously. However, we investigated optimizing an SNN classifier to increase energy efficiency. Due to various issues during the training with surrogate gradient and the evolutionary optimization, we discovered a method to define a network where the weights are precalculated based on Equation 6.3. Thus, we first demonstrate the encoding ability and the limitation of a single R&F layered network. At the same time, we focus on the two-layer architecture and the comparison to the two-dimensional FFT. Our last experiment uses actual measurements to calibrate the system and improve the angle estimation.

6.2.1 Single Target Scenario

The first experiment focuses on the first R&F layer within the architecture shown in Figure 6.1, where we investigate the information content of the output spike events. As a baseline method, we use here the Fourier transformation approach. We recorded measurements of an object moving on the azimuth angle from $-80°$ to $80°$ in a 3.6 meter distance. Each angle measurement was repeated a minimum of two times. We used a rope between the sensor's center and the object to ensure the same distance.

Setup

The radar setup uses only one transmitter of the radar sensor with a bandwidth of 607.7 MHz for a ramp duration of 20.48 μs and a total of 512 sample points per chirp. In the given experiment, we assume that we only use 2 out of 16 receive antennas to detect the single target. We sort the channel data from the radar sensor into the order of the physical layout to ensure that we compare the two consecutive channels. As data preprocessing, we apply a DC offset suppression by subtracting the mean of the sample points. The second preprocessing step is a Hanning window which reduces the effects of the discontinuities at the boundaries of the chirp signal. The windowing is necessary because the current setup receives a single chirp instead of a continuous input.

As a baseline method, we use the state-of-the-art method to calculate the range by a Fourier transformation. The method is basically already explained in Chapter 2.1.1 with an extension of calculating the phase difference between two consecutive channels. Additionally, we calibrate the method on a measurement with zero angles at a known distance.

We implemented 14 R&F neurons between the range from 3.5 to 5 meter distance, where we select the closest neuron to the target. The neurons use the phase-dependent spike mechanism and no reset. In the given example, we only investigate a single chirp such that the network provides a higher activity for these short sequences. However, this approach presents a drawback when dealing with longer sequences, and it leads to the phenomenon of exploding state, causing instability. A reset operation becomes imperative to address this challenge in the context of continuous application across multiple chirps.

In order to decode the first layer of R&F neurons, we calculate the mean temporal difference between the two spike trains. Therefore, we convert the event matrix into a sparse representation, which contains only the indices and times of each event. The mean of the smallest temporal difference of each event between the test and reference spike trains contains the relative phase of this frequency. The distance between spike events depends on the resonance frequency of the neuron and thus limits the maximum detectable temporal difference between spike events. This limitation prevents a significant shift in the relative phase due to unbalanced events in the spike trains, which can occur due to noisy input data. The mean temporal difference Δt of the neurons with the resonant frequency f forms the phase difference θ between the spike trains as defined by:

$$\theta = 2\pi f \Delta t. \tag{6.5}$$

We calculate the angle of the object based on the following equation:

$$\alpha = \arcsin\left(\frac{\theta}{2\pi d}\right) \tag{6.6}$$

Also, we have to calibrate the decoding method to compensate for phase imbalance between the channels. So, we define the calibration vector within the temporal domain by extracting the mean spike distance of a scenario with a single target at 0°. We then apply the temporal shift to the estimated mean spike distances for the different angles instead of shifting the phase.

Results

In the following experiment, we do not report the range sensitivity of the R&F layer because we have already explained it in Chapter 3.4. Figure 6.3 shows the expected versus the predicted angles of the FFT and R&F angle detection by investigating the phase differences between the channels. Notably, the frequencies of the corresponding neuron do not exactly match the FFT processing. Due to the calibration, the angles are moved toward the ideal positions. We can observe that the predicted angles of the FFT and the R&F method are similar except for the higher variations towards the boundaries.

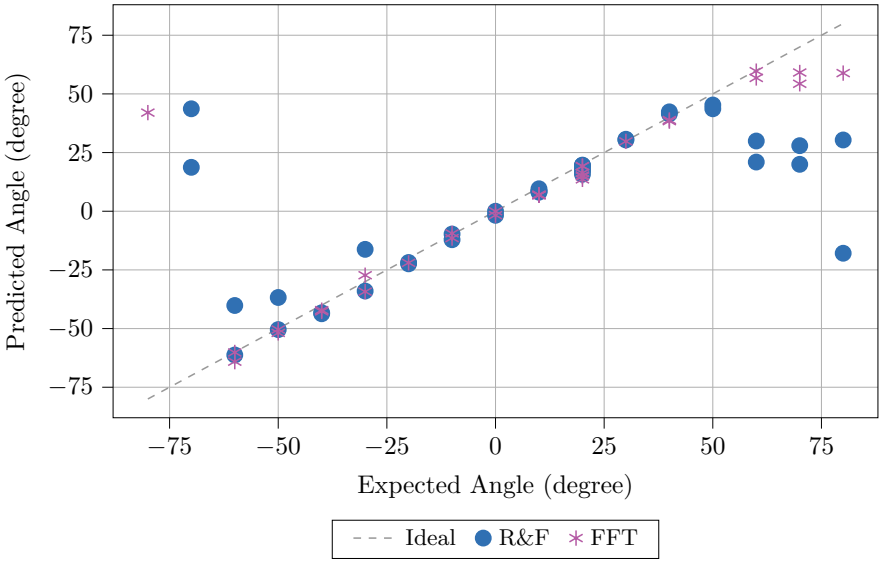

Figure 6.3: Single target angle estimation: Angle estimation of a single target at a constant distance with R&F and FFT processing. The estimated or calculated phase difference between two receive channels defines the angle of the target. Both methods are calibrated on the zero angles. The diagonal line represents the ideal point of the angle estimation.

Especially broader angles than $\pm 50°$ are less reliable than predictions close to zero degrees. Also, for $-80°$ there is not a single prediction of the R&F method because the object's echo leads to a neuron stimulation below the threshold. A threshold decrease would make the object visible in this visualization but generates a low spike count and therefore has a high phase error due to the discrete simulation of the R&F neurons. In general, the precision of the angle increases with more spike events because the predefined step size of the simulation introduces an error that the mean of the spike time differences can mitigate.

We use the Root-Mean-Squared-Error (RMSE) as a metric between the expected and predicted angles without the undefined ones. The RMSE is 22.31 and 28.73 for the FFT and R&F methods. The difference mainly occurs due to the boundary areas of the R&F angles because an angle precision is lower for broader angles. At $-30°$, we have an unexpected outlier where we expect the reason to be the imperfect measurements during the data recording because the FFT results are also not ideal.

Often, the spike count is an estimate of the energy consumption because sparse temporal events reduce the communication overhead and the necessary updates of the neurons. Here, we use the first R&F layer as an encoding and as the time-to-frequency conversion method, which means that the input continuously changes and updates are necessary at each point in time. However, the spike count is an important metric if the preceding layers use the output spike events. On average, 5.9 spikes are emitted per neuron for a time series of 512 sample points, and the total population emits 82.66 on average for

all angles. A selected neuron that represents one target generates 10.94 spike events on average. Noteworthy is that the number of events highly depends on the threshold, and it needs to be selected appropriately for the specific problem.

Outcome
Here, we have demonstrated the very first proof of concept that the R&F neurons of the first layer are already capable of detecting the angle of a single target. The resolution of broader angles is lower than the Fourier transformation method but can be optimized with suitable parameter tuning. Also, we have shown how to calibrate the spiking version with an artificial synaptic delay. However, when multiple targets at the same distance with different angles exist, the phase difference between the two channels cannot provide enough information to differentiate the two targets. This method is designed for two receiving channels. A further extension of the channels would also increase the complexity of the decoding algorithm without keeping the information within the spiking domain.

6.2.2 Multi-Target Angle Estimation

With the previously demonstrated single target angle estimation method, with only the first R&F layer, a scenario with multiple targets per distance is not resolvable. However, there are applications where such a method is adequate but not within the automotive radar because scenarios like on a parking lot or the street lead to multiple targets with the same distance at different angles. Hence, the following experiments concentrate on the solution of the multi-target angle estimation with the architecture shown in Figure 6.1.

Setup
In this experiment, we use simulated data with the previously mentioned radar simulator with different settings to demonstrate the flexibility of the proposed algorithm. We simulate one transmitter with a bandwidth of 275 MHz, a ramp time of 26 μs, and 512 sample points. The 32 receive channels are already in the right order, and reordering to the physical location is unnecessary. The data preprocessing is the same as in the previous experiment with DC offset suppression and a Hanning window function. In the simulated data, we have not included any channel imbalance. Therefore, we do not need to calibrate the signal processing. The simulation consists of two targets with a constant RCS moving towards and away from the sensor with $\pm 20°$. The RCS simplifies and would, in reality, change with the distance, as explained in Chapter 2.1.1. The baseline method is the signal processing with two FFTs across the time steps and the channels, as explained in Chapter 2.1.1. The second FFT also uses 32 channels, and we apply no further zero padding.

Here, the first layer of the R&F architecture consists of 64 neurons ranging from 2 to 15 meters, and the second layer of 64 neurons spaced linearly between $\pm 90°$. The resonance frequency of the second layer is the same as the first layer, and the complex weights are precalculated on the linear spacing of the angles. All neurons use a damping factor of 2, but a change does not significantly affect the results. The threshold of the

first layer is set to 4 and for the second layer to 1.5. Again, we do not use a reset after a spike event in both layers, but the individual chirps are processed with a clean initial network.

Results

In Figure 6.4, we visualize some selected example results to explain our observations further. The first observation is the capability of the encoding layer to convert the temporal signal into the frequency components, where the phase-dependent spike mechanism realized by a Heaviside step function has the advantage of removing noise and setting the focus on the targets. However, a high threshold has the drawback of an increased likelihood to target loss.

In Figure 6.4 (a), we show the mean magnitude of the channels by accumulating the spike events across the time dimension. Comparing the two approaches, we identified that the FFT shows a lower magnitude for the more distant object while both targets have the same RCS. Therefore, a closer target is more likely to be detected than a further one because the RCS depends highly on the target's distance. In the case of the R&F neurons, the magnitude of the second target is even higher because the number of spikes correlates with the resonance frequency and is not affected by the amplitude of the neuron state.

The next visualization in Figure 6.4 (b) highlights the differences in the angle estimation, where we limit the representation to a cross-section of the azimuth angle at a distance of 9 meters. The two targets at the same distance are separable by their angle of $\pm 20°$ with all shown methods. We decode the angle of the R&F method by accumulating the spike events over time. The biggest difference is the sparsity of R&F processing, where the threshold removes the noise and reduces the information compared to the FFT. For the given examples, the accumulated events lead to two out of 32 values and therefore reduce the number of output information by a factor of 16. Here the object at $+20°$ is slightly smaller than the counterpart by only a single spike event which more time steps could mitigate. However, the R&F method is also not ideal because the threshold greatly influences the detectability. A dashed line shows an example of a reduced threshold where the targets can still be detected, but other unwanted artifacts are introduced, comparable to those in the FFT. One problem with a higher threshold is that closer targets are less likely to be detected because they generate fewer spikes due to the phase-dependent spike mechanism that emits spikes with the same frequency as the neuron's resonance. Consequently, compensation for this effect could use a distance- or frequency-dependent threshold value.

Finally, we show in Figure 6.4 (c) and 6.4 (d) the two scenarios as range-angle maps, where the normalized accumulated spike events of the second R&F layer. The two targets are perfectly separable for all scenarios, but the detectability of closer targets is difficult with a static threshold. Because we do not use the reset of the neurons, the number of spikes is high. We calculate the mean spike events for a single chirp event by summing across the time steps and the neurons, accumulating to 5183.4 events, where the first layer consumes 96.55% of the spike events and the second 3.45%. Therefore, the first and second layers have a mean spike count per neuron of 2.44 and 0.09, respectively.

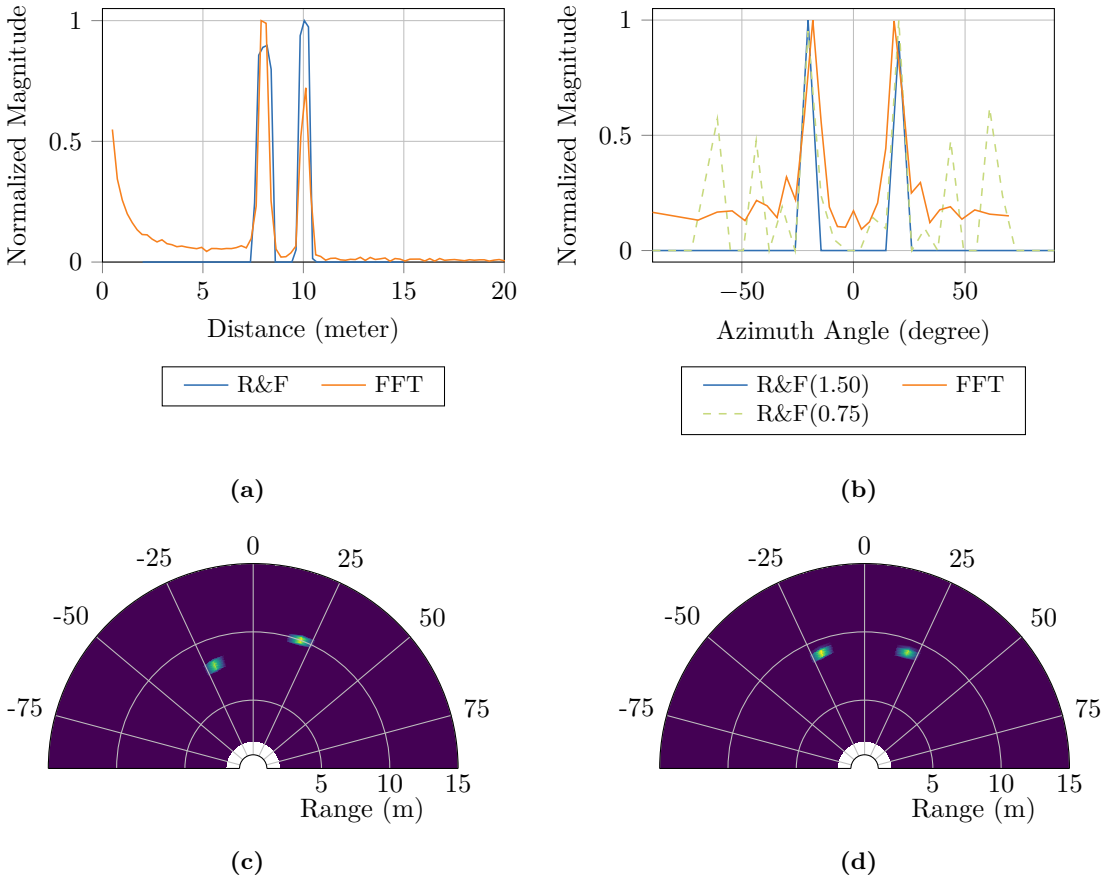

Figure 6.4: Multi-target angle estimation with two targets: Two examples extracted from a two-target simulation where the targets move towards and away from the sensor with an angle of ±20°. In the first example, shown on the left side, the targets are at 8 and 10 meter distances, and the second example, on the right side, demonstrates both targets at 9 meters. The RCS is constant for the complete simulation sequence and does not reduce with the distance. In (a), we show the mean representation across the channels of the accumulated spike events of the first layer in comparison with the FFT, and (b) shows a cross-section of the distance 9 meters where both targets are located at a different angle. Additionally, we visualize two different threshold settings of the second layer that introduces some artifacts. The bottom figures, labeled as (c) and (d), show the range-angle maps in the polar representation of the two selected scenarios by accumulating the spike events over time. The normalized amplitude is depicted by the color intensity as depicted by the corresponding range plot, where further targets lead to more spike events. The two targets at the same distance have different colors because the spike count differs only by one spike event.

Important to see the spike count in relation to the 16384 sample points for all signals of the channels, where the total number of spike events consists of about three times fewer points than the input signals.

Additionally, we have investigated the damping factor that leads to a damped increase in the neuron amplitude and a decrease when the frequency is absent. Therefore, a modification of the damping factor introduces a necessary adjustment of the threshold to compensate for the lower or higher amplitude. However, as expected, the damping factor has no further effect on the detection of the two objects because the frequencies of the objects exist for the complete chirp. The importance of the damping factor increases for concatenated chirp scenarios and continuous R&F applications.

Outcome

We have shown in the given experiment that two layers of R&F neurons are able to differentiate the angle of objects within a predefined field of view. The comparison with standard algorithms like the FFT is challenging because the parameter space of the R&F is not yet optimized. Further improvement could increase the reliability and the object's separability. However, we have shown as the first the concept of using a fully spiking angle estimation method with R&F neurons and complex synapses. The benefits are not yet clearly extracted but promising is the communication reduction to events instead of floating point values and the possibility to apply this method in a stream processing setup. Also, the modification of the number of sample points does not change the memory footprint because the dimension of time is continuously used.

6.2.3 Calibration

An important but smaller experiment is the calibration of the two-layer angle estimator because the channels are imbalanced in the real world and therefore have small amplitude and phase deviations, reducing the angle estimation capabilities. Thus, we investigate the improvement of the angle estimation through the FFT-based calibration of the complex weights in the following experiment. We have to reorder the data before processing since the FFT approach needs the physical alignment of the channels.

Setup

As in previous examples, we prepare the recorded data with DC suppression and the Hanning window. Instead of simulated data, we use recorded data that contain a single target at different azimuth angles with a constant distance of 3.6 m. The benefit of recorded data is that we additionally take into account the non-linear effects of the antenna design, but the data are limited to 16 channels. The network architecture is the same as in prior experiments with different settings. We investigate the distance between 2 and 8 meters with a constant damping factor of 2. We select the threshold such that we see a spike response in the calibrated and uncalibrated version that leads to 2.5 and 0.5 for the first and second layers, respectively. Also, we demonstrate the capabilities of interpolation between the angles by using more angular neurons than existing channels.

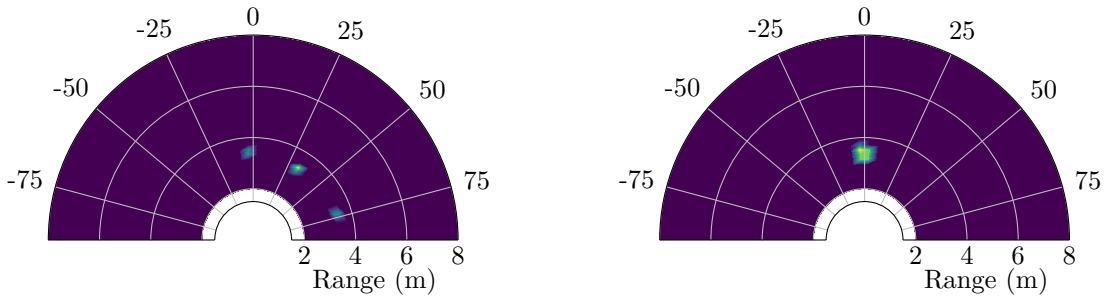

Figure 6.5: Uncalibrated vs. calibrated angle estimation: comparison of the uncalibrated (a) and calibrated (b) polar representation of measured radar data. The broad target around zero angles is the reason for the linear scaled angles between ±90° where the closest angles are ±2.9°. In the uncalibrated case, spike events are spread across the azimuth direction, reducing the maximum amount of spikes to 2. In the calibrated version, the spikes are focused on two consecutive peaks, which lead to a maximum of 5 events.

The data employ 16 channels, whereas both layers use 32 neurons in range and angular dimension.

Results

The example results in Figure 6.5 show the improvement of the angle estimation by using an offset calculated by the two-dimensional FFT. In the uncalibrated version, the object is spread across multiple azimuth positions, indicating multiple objects at the same distance. These ghost targets are the reason for false relative phase rotation of individual channels due to physical effects. This azimuth spread reduces the number of output spike events to 7 compared to 27 because the number of matching spikes is reduced heavily, and the likelihood of damped oscillation increases. However, we can observe that there is no negative effect on the range information in both variants. After the calibration, the object is shifted to the zero angles, as depicted in Figure 6.5 (b), and the ghost targets disappear. We can observe a spread of spike activity in the range and angle dimension, which is a consequence of the interpolation of the 16 channels to the 32 linear scaled angles. Therefore, the information is distributed to the ±2.9° angle instead of the exact non-existing zero angle.

Outcome

Here, we have demonstrated the calibration improvements by calculating the calibration vector based on the two-dimension FFT. Combined with the first single target experiment, we have shown the two calibration realizations by introducing an additional synaptic delay or applying an offset to the complex weights. However, adjusting the complex weights is far more cost-efficient because we do not need additional delay parameters, and we can remove the delay processing. An additional programmable delay element in analog and digital hardware implementations increases the complexity. However, the

shown calibration approach is enough for the proof-of-concept of the R&F neuron angle estimation. Additionally, we have demonstrated the interpolation of the angles by a factor of two and realized that an application-dependent angle scaling would increase the algorithm's efficiency.

6.3 Discussion and Summary

In this chapter, we have demonstrated conceptual proof to answer the research question 4 that spiking neurons can estimate radar signals' arrival angle. The R&F neurons act simultaneously as encoding and processing neurons, where the spatio-temporal data are sparse binary events that carry information about frequency, phase, and amplitude. Normal synaptic connections fail to differentiate the physical source of an event, and only the introduction of the complex weights solve this issue. This research can be separated into two fields: SNN and echolocation. In biological systems, echolocation is performed with a low power budget with spiking networks. Therefore, the field of bio-inspired systems emerges in applications using the angle, where the research mainly focuses on robotics applications like head position localization [259] or steering angle estimation [260]. However, the angle-of-arrival in radar systems means the localization of high-frequency echos. Currently, there are three advanced traditional algorithms to solve the angle estimation. The most famous one is the FFT across the channel dimension. MUSIC [82] and ESPRIT [83] are more computationally complex, outperforming the FFT, but prior knowledge of the number of components is necessary. These methods have one thing in common: they can work with various transmitter and receiver combinations, which is currently a limitation of the proposed angle estimator. The increasing number of virtual antennas is the limitation of the Rayleigh Criterion [60], which poses a limitation to distinguishing two closely spaced objects.

Currently, deep learning methods also solve the angle estimation problem, where [72] provides a good overview. The basic components of the R&F implementation support machine learning, but the spiking methods suffer during training. The training of R&F with a surrogate gradient is very unstable due to the oscillating effects, and the temporal dimension increases the likelihood of vanishing and exploding gradients. However, one future research would be the evolutionary optimization of the proposed architecture. We have only analyzed whether a LIF population with real-valued synapses can be optimized using an evolutionary method that failed. A trained NN is superior to traditional methods but also needs an extensive and carefully created dataset, as depicted by [261].

Additionally, the parameter choice is challenging and highly depends on the application and the trade-off between the targeted low echo detection and noise suppression requirements. The additional setup effort enhances the event sparsity and reduces the computational effort of the subsequent layers. Each layer acts as an additional information filter and focuses on specific ranges or angles. Similar to the FFT, the R&F underlies the Heisenberg-Gabor uncertainty principle [262], which states that the position and momentum of a particle are impossible to determine simultaneously.

Nevertheless, the R&F method continuously represents the time and frequency by generating spike events without storing temporal events in an additional memory. Also, subsequent layers can be attached in a streaming pipeline with a low overall latency due to the temporal processing and communication of the binary events instead of the complete output map. A state-of-the-art 2-dimension FFT implementation, like [263], stores the intermediate results of the first FFT in memory and therefore reduces the necessary silicon area but increases the latency of the detection. Suppose we assume the single Fourier transformation can generate one frequency bin per cycle. In that case, a sequence of length 512 needs at least 1024 clock cycles for the 2-dimension FFT without considering the initial latency and memory read/write timings. In the case of the R&F neurons, the final result of the angle estimation is already present after 512 time steps, which we assume to be equivalent to the clock cycle. We also neglect the initial delays. However, due to the direct interaction between the two layers, we have an early, less accurate prediction of the final result. The reduction of the latency and earlier prediction is an essential improvement in fast and dynamic environments like automotive applications.

Interestingly, we could combine a temporal, rate, and population encoding scheme by the R&F neurons. Depending on the relation, the temporal information between spike events carries information about the relative phase or even the signal's amplitude. A rate only occurs over time and indicates how long a frequency exists within the signal, and the population informs about the existence of the resonance frequency. Therefore, the first layer acts as a frequency-selective filter, and the second layer performs the processing of the temporal relation between the spike events. The complex weights perform additionally to the processing of the angle a mapping of the memory to the physical antenna layout and reduce the necessary memory rearranging.

The first layer of R&F neurons is crucial because these neurons differentiate high or low-amplitude signals. Frequency leakage also appears with R&F neurons, and the threshold separates the spiking and non-spiking regions while removing noise. Therefore, a high threshold increases the likelihood of losing low-reflective objects. A low threshold reduces the separability of close targets due to frequency leakage and the loss of quality. The accumulated spike events generate a plateau around the target frequency, and further research with approaches like adaptive dynamics or inhibitory connection are promising to improve the first layer.

Overall, this chapter answers the RQ 4 by demonstrating the end-to-end application of angle-of-arrival estimation. So far, this application is not fully evaluated yet, and future improvements are possible to realize a low-power bio-inspired angle estimation. Until now, we can classify the two categories of optional improvements either by optimizing the algorithm or by an extension. Optimizations range from enabling the capability to process a chirp sequence as a continuous stream and even using the inter-chirp interval knowledge. Also, adjusting the threshold depending on the range increases the response to lower signal amplitudes and introduces a fully spiking calibration method. Distant objects generate more spike events because the spike frequency depends only on the neuron's resonate frequency, but the target's RCS is also lower, which leads to the question of how the RCS and the spike rate correlate.

Additionally, there are possible optimizations in terms of hardware implementation because a mixed implementation seems to be most promising by an always active first layer and an event-driven second layer for lower power consumption. However, this is a completely independent research area and allows exact power consumption measurement of specific applications. Possible extensions are inhibitory connections to prevent frequency leakage and increase the focus, similar to CFAR algorithms. Also, using the continuous processing and extending the system with tracking algorithms to extract the velocity information and detect the scene better. In the current implementation, we assume only a single transmitter, but the angular resolution can be further improved with more complex modulation schemes like time- or phase-division-multiplexing. Such an increase in virtual antennas also increases the computation complexity of the algorithm. Besides, comparing detection performance, computational complexity, power consumption, and latency increases the acceptance of such bio-inspired angle estimation methods.

It was demonstrated by [72] that deep learning methods outperform current state-of-the-art angle estimation algorithms with increasing computational costs, and [264], [265] show the high energy efficiency of SNN, which motivates to invest in research about the angle estimation based on spiking neurons by using the R&F neurons as encoding and processing element. This work demonstrates the first proof of concept on real measurements with improved angle estimation due to the FFT-based calibration.

7 Summary and Conclusion

This work combines the three diverse research fields of radar signal processing, Artificial Intelligence (AI), and neuroscience. At first view, these fields might contradict since the radar system is designed for frequencies around 77 GHz while biological neurons operate only in the Hz range. However, the general concept of radar processing has considerable similarities to the evolved echolocation of bats that motivated the realization of SNNs for radar signal processing while applying AI methods like BPTT for the optimization of the weights. To further reduce preprocessing, we developed a direct transformation of the sensor's signals to spatio-temporal spike events. The encoding method and subsequent SNN architectures take advantage of sparse, asynchronous, and spike-based communication.

Chapter 1 provides a compelling motivation for the work and highlights the key research questions. Subsequently, in Chapters 2.1 and 2.2, we establish the essential background on radar and SNNs respectively. Moving on to Chapter 3, we conduct a comprehensive survey of current state-of-the-art methods to encode real values into spikes and introduce the R&F encoding. The R&F encoding serves as the foundation for subsequent applications, which employ various spiking methodologies to compare encoding methods and emphasize the unique features of the frequency-selective encoding. Notably, these applications demonstrate the usage of SNNs in this context for the first time, as far as the authors are aware.

Chapter 4 showcases two fundamentally different concepts for classifying radar interference using traditional and spike-based methods. In the following chapter, we analyze the conceptual differences between the LIF and R&F encoding through the investigation of a phase detector application. To validate our findings, we implement the spiking phase detector with LIF neurons on analog hardware. In the final application described in Chapter 6, we explore the capability of SNNs to detect the angle-of-arrival of multiple objects in a three-dimensional environment.

The following sections provide a sophisticated summary of the individual chapters and the limitations of SNN processing.

7.1 Background

We separate the background into two stand-alone chapters covering the basic knowledge and the state-of-the-art about radar and SNNs. First, Chapter 2.1 focuses on the radar sensor, from the theoretical concept to the signal processing pipeline. Before we derive the equation of the instantaneous frequency of a single target, we explain the general concept of FMCW radar systems. Afterward, we demonstrate the current standard signal processing pipeline consisting of multiple Fourier transformations focusing on range and

angle estimation. In the last part, we emphasize the application of radar interference detection by deriving the IF equation. Since we use simulated data, this chapter covers the theoretical background for synthetic data generation.

The second background chapter focuses on SNNs by introducing the properties of an action potential, called spikes, which is the primary communication element in SNNs and the main difference to earlier NN generations. Thus, we explain the Hodgkin-Huxley neuron model, which is the basis of each type of neuron, and the authors were awarded the Nobel Prize in 1963. Since the Hodgkin-Huxley model encompasses a high dynamic, it is computationally too expensive to be viable used in industrial applications. Therefore, we introduce the LIF and R&F neuron models that build the basis of this work. We also describe the synaptic models, the properties of synaptic polarity, and the topological properties of spiking networks relevant to our experiments. This thesis uses either analytically calculated weights or weights that have been trained using gradient descent optimization with pseudo gradients. Thus, we present the background of current state-of-the-art learning methodologies and describe the details of the gradient descent training. We introduce the idea of spike and synapse sparsity for architectural and energy optimizations.

7.2 Information Encoding

Since encoding information into the spike domain is crucial for efficient processing with SNNs, we demonstrate different methods in Chapter 3. The spike encoding combines the task of pure transformation into spikes and the lossy information compression to the necessary information. Generating many spikes can overwhelm the preceding network, while a low spike count provides not enough information to process the data. Additionally, we identify two variants to encode sequential data: frame-based and stream-based.

Encoding methods like pure count rate or TTFS encoding are frame-based approaches since they convert the information into a spike train with sub-sequences, enabling a representation of the temporal or the rate-based spike trains. Thus, such frame-based encoding methods are not suited for continuous applications but are applicable to image-based input data with a low update frequency. In the signal-processing domain, these frame-based approaches are more powerful in combination with preprocessing steps like the Fourier transformation.

The stream-based approaches convert continuous signals without the introduction of fine-grained sub-sequences, enabling an immediate encoding of one-dimensional sequences. This group contains methods like the population code, where each neuron is responsive to specific amplitudes, or filter-based methods like BSA or HSA. Due to the sinusoidal character of the radar IF signal, we propose the current injection encoding with R&F neurons in Chapter 3.4.

The R&F neurons convert a one-dimensional analog input stream into a spatio-temporal spike response, where the correlations of spikes carry the information about the signals' amplitudes, frequencies, and phases. We demonstrate that the resonant behavior of the R&F neurons realizes a frequency-selective property that can be used with spike- or

signal-based inputs. When the resonant frequencies of the neuron match the present input signal components, it increases the magnitude of the neuron oscillation and fires when reaching the threshold potential. We present different spike functions and reset methods of the R&F that we further investigate in the application chapters. Additionally, we analyze in Section 3.4.4 the similarities of the R&F encoding with the Fourier transformation.

7.3 Applications

For the analysis and investigation of the proposed encoding method, we evaluate the applications of radar interference detection, phase detection, and angle-of-arrival estimation in Chapters 4, 5, and 6. These applications utilize different signal-processing properties of sequential data.

7.3.1 Interference Detection

In the first application in Chapter 4, we investigate the realization of an SNN for interference detection. Interference is a disturbance of the receiving signal, appearing when the frequency course of multiple sensors coincide. We develop two methods to identify interference: the prediction and the classification approach. Therefore, we define interference as an anomaly of the radar signal, and throughout all experiments, we use simple state-of-the-art amplitude thresholding as the baseline method. The interference detection answers the RQ 3 and demonstrates an example of the RQ 1.

The prediction approach is a semi-supervised method where the anomaly score is defined by the deviation of a predicted to the real signal. The network learns to predict the features from the normal signal during supervised training. As the underlying architectures of the predictor, we have investigated LSTM, GRU, dense, and SNNs with current injection or temporal contrast encoding. Unlike traditional ANNs, SNNs have not mastered extracting the features from the time series.

Pattern classification is a fully supervised architecture explained in Section 4.2.2. A classifier either predicts the normality or the outlier, equivalent to interference. The classification takes advantage of the proposed R&F encoding, which is difficult to train. Therefore, we perform an initial architecture search with traditional and spiking networks utilizing different encoding methods. Additionally, we demonstrate architectures with an STFT-preprocessing step, reaching the highest performance in combination with a CNN. For the spiking architectures, the R&F encoding performed better than the current injection or temporal contrast encoding. Nevertheless, the phase-dependent R&F neuron does not provide any benefits because only the frequency changes are essential in the interference detection application.

Furthermore, the interference application covers the simulative comparison of the different reset and spike functions. Also, we prove that a minimum spike activity is necessary for the classifier by implementing spike regularization and synaptic sparsity. The spike regularization is vital for reducing the number of events by keeping the accuracy stable, and wrong parameter selection leads to worse classification accuracy. However, the stability of the network during the pruning of synaptic connections indicate that spiking

networks are less prone to synaptic connection failures, making the third-generation networks highly attractive for safety applications.

All SNNs are designed to be implemented directly on the input stream by using the inherent features like the temporal memory in the membrane potential. Thus, we prove that an SNN classifier with the R&F encoding uses the local state of the neurons to remember the short-term temporal correlation between spike events since interference leads to a sequential stimulation of different frequencies.

7.3.2 Phase Estimator

In the previous section, we select a task where the temporal information of a spike carries the information of the signal. Now, in Chapter 5 we show the realization of a spiking phase detector that uses the signal's phase difference by the correlation of the exact spike events.

Therefore, we implement a four-neuron system with two encoding and decision neurons. The encoding layer converts the input signal into spikes where the spike timing difference is proportional to the signal's phase. The second layer decides if a signal is within or outside the expected limits depending on the selected weights. We separate the experiments of this application into the analysis of encoding and phase detection. Thus, we compare the LIF and R&F encoding to extract the phase of a single- and multi-tone signal. The inherent frequency selectivity makes the R&F neurons superior to the LIF for multi-tone and noisy signals.

We implement the LIF phase detector on analog hardware and validate the simulations with hardware measurements. Additionally, we show that the functionality of the implementation could be monitored by observing the spike output. One limitation of the current systems occurs during the positive phase shift. When the test signal occurs before the reference signal, the inhibitory connection is useless, and the neuron indicating the out phase generates false spike events. We propose multiple methods to overcome this limitation.

7.3.3 Angle-of-Arrival Estimation

Chapter 6 covers the angle-of-arrival estimation application, utilizing the amplitude, frequency, and phase information. The current state-of-the-art algorithm is presented in Chapter 2.1, which we consider as a baseline for the comparison.

The proposed architecture uses the inherent properties of the R&F neurons, where the first layer converts the radar signals into its frequency components and the phase by the timing of the spike. The second R&F layer converts the spike timings to the angular position, and the temporal integration of the output spikes generates a range-angle map.

In the first part of the chapter, we explore the capabilities of the encoding layer, which is demonstrated to be sufficient to detect single-target scenarios but struggle with more than one target at the same distance. We compare the single-layer results with those obtained by a classical Fourier transformation-based approach by analyzing the relative phase difference of real measured data. Afterward, we compare the azimuth angle

estimation of the Fourier approach and the two-layer R&F architecture for a multi-target scenario. Since the electronic components of a radar MMIC are not ideal in the real world, a phase shift exists between the receiver channels. Thus, we propose a method to calibrate the R&F angle-of-arrival estimator with real measurements.

The angle-of-arrival estimation provides an answer to RQs 1, 2, and 4 by the combination of a suitable encoding of the time-domain radar signal with a fully spiking architecture. With this proof of concept, we show that spiking neurons can detect objects in the three-dimensional space. We observe advantages like noise suppression, sparse spike events, classification speed, and a more efficient data handling since the complex reorganization and shuffling of the data in memory is not required. Still, further research is necessary to apply this in an industrial application, but we open a new field of SNN processing.

7.4 Limitations

This work presents three spiking applications using R&F encoding. Overall, our experiments indicated that the R&F encoding is valuable for sinusoidal signals and that the networks outperform other algorithms by their sparsity and faster classification. Nonetheless, the following limitations must be considered.

The encoding method significantly impacts the overall power consumption of the event-based SNNs by selecting only necessary features and increasing the information sparsity. The advantages of SNNs disappear for rate-based encoding, and ANN implementations are favorable. The proposed R&F encoding combines temporal and population coding with the information in the ISI. However, this encoding may not apply to other applications, like the classification of images or clock-based signals.

The third-generation neurons consist of complex dynamics providing a higher computational performance while reducing the number of necessary neurons. This advantage comes with the disadvantage of a complex design space due to many neuron parameters. In this work, we applied parameter sharing to reduce the complexity, but in a biological system, we expect that each neuron is unique, achieving a higher computational capability. Examples of parameter sharing are the threshold potential and membrane time constants.

The industrial relevance of the spiking networks mainly depends on the reliability of the optimization. As widely known, training methods like gradient descent are the standard in ANNs. However, the adaption of gradient descent to support SNNs comes with drawbacks like instability and neglecting neuronal dynamics. Since spiking neurons take advantage of the temporal dimension of the input, the training has to use BPTT. Therefore, the training of SNNs suffers from dead neuron problems, and these deep unrolled networks are susceptible to vanishing and exploding gradients. Since gradient-based optimization is not designed for the high dynamics of the spiking neurons, it is less effective. However, the accuracy often reaches state-of-the-art performance compared to biological methods like STDP.

Since we have fabricated the phase detector, we can provide the system's energy consumption. The other two applications are not implemented on neuromorphic hardware,

preventing a valid energy measurements. Also, neuromorphic simulations, designed to simulate many neuron models, are not comparable to highly optimized accelerators for algorithms like the FFT. For a valid comparison, these networks must be implemented in dedicated analog or digital hardware. Furthermore, it should be considered that the power consumption depends not only on the spike count, which is often used for energy estimation, but also on the network architecture, hardware properties, neuron types, synaptic models, and the appearance of asynchronous events.

8 Outlook

The field of SNNs provides an enormous research potential because the exact functionality of the brain is still unknown, and further research in understanding the brain is necessary. Also, investigations about using biologically inspired NNs provide a better understanding of the functionality and can prove the theoretical ideas. The outcome of this work opens several directions for future research. The event-based nature of SNNs offers the advantage of reducing power consumption because only the processing of an event consumes energy. In the electric vehicle industry the reduction of the power budget means a significant price and weight reduction of the battery system while achieving the same travel range.

Stable and reliable training is most relevant for the success of SNNs in industrial applications. Today, the community mainly uses backpropagation with pseudo gradients, which is neither biologically explainable nor stable. Recently, Geoffrey Hinton proposed the forward-forward algorithm [266], which has similarities to predictive coding [267]. The algorithm replaces the backward passes during backpropagation with a forward pass, which uses an objective function for positive and negative data. The goal is to maximize the objective of the positive pass and minimize the negative pass. Additional to the improved learning for resource-constrained systems, the individual layers are not necessary to be differentiable, making it more applicable to SNNs. Another direction is the anytime classification which means that the neurons should continuously classify the input instead of windowed classification such that input changes directly affect the prediction.

Another central aspect to consider from an industrial perspective refers to functional safety. Our investigations have demonstrated that SNNs exhibit a reduced susceptibility to the pruning of synaptic connections compared to ANNs. Nonetheless, this finding alone does not definitively address the level of safety inherent in spiking algorithms. Specifically, we are concerned with the probability of failures during the deployment of neural networks, as opposed to potential effects resulting from variations in dataset distributions, class imbalances, and other training-related challenges. Thus, there are different research directions necessary. Firstly, it is imperative to evaluate the generalization capabilities and failure probability of SNNs. Secondly, we must investigate the stability and latency of hardware deployment, particularly in the context of analog implementations. In this regard, the generation of spikes is profoundly influenced by the stability of the voltage source, which could be compromised by the simultaneous spike generation within a larger neuron population.

We observe that the sparsity of the events increases with the depth of the neuron layer, which means there are fewer spike events in the high hierarchical layer of the network. Therefore, a mix of analog and digital hardware would be an efficient solution.

The analog hardware contains the lower layers, including the encoding, and the upper layer is implemented on digital hardware to save area by an event-driven architecture. Therefore, analog neurons and encoders like the R&F with the phase-dependent spiking need to be implemented by converting the continuous to discrete spike events and a digital event-based neuromorphic processor. Such architectures would provide an indication of the real power, latency, and area measurements for a fair comparison with highly optimized algorithms like the Fourier transformation.

For radar applications, we see further potential in a smart and dynamic adjustment of radar sensing, focusing on the most important locations via beam-steering. One solution to achieve a cognitive radar is the signal processing with R&F neurons by estimating the range-angle map and tracking the objects. However, the angle estimation can further be optimized by continuous observation of phase drifts to automatically adjust the phase calibration or use inhibitory connections to reduce frequency leakage across neurons.

This work lays the foundation for directly converting the radar signals to spikes and the subsequent signal processing with an SNN. The research areas of understanding the nervous system and the signal processing with SNN-based algorithms can positively influence each other. This work uses biological observations to deploy spiking systems with a suitable encoding, proposing the usage of spatio-temporal spikes instead of pure rates or time encoding. Nevertheless, the author enthusiastically looks forward to the promising future of SNNs, which will help to improve our understanding of the nervous system and unlock their enormous potential in various industrial applications.

A Interference Classification Architectures

The following attachment is part of Chapter 2.1.3 and contains information about the individual network architectures used throughout this chapter. Since the individual architectures with the R&F encoding vary slightly, we only report the default values and do not include the full list of all architectures.

- LSTM
  - Layer 0: LSTM (input_dim=1, hidden_size=28, activation=ReLU)
  - Layer 1: LSTM (input_dim=28, hidden_size=2, activation=ReLU)
- LSTM-Dense
  - Layer 0: LSTM (input_dim=1, hidden_size=28, activation=sigmoid)
  - Layer 1: Linear (input_dim=28, hidden_size=2, activation=ReLU)
- CNN-Dense
  - Layer 0: Conv1d (in_channels=1, out_channels=4, kernel_size=10, stride=8, padding=0, activation=ReLU)
  - Layer 1: Linear (input_dim=28, hidden_size=2, activation=ReLU)
- CNN-Dense
  - Layer 0: Conv1d (in_channels=1, out_channels=4, kernel_size=10, stride=8, padding=0, activation=ReLU)
  - Layer 1: Linear (input_dim=112, hidden_size=2, activation=ReLU)
- STFT-CNN-Dense
  - Preprocessing: STFT (n_fft=256, hop_length=32)
  - Layer 0: Conv2d (in_channels=1, out_channels=8, kernel_size=9, stride=(6,2), padding=0, activation=ReLU)
  - Layer 1: Linear (input_dim=168, hidden_size=16, activation=ReLU)
  - Layer 1: Linear (input_dim=16, hidden_size=2, activation=ReLU)
- STFT-LIF
  - Preprocessing: STFT (n_fft=256, hop_length=32)

  - Layer 0: LIF (in_features=129, out_features=22, tau_syn_inv=7142857.0, tau_mem_inv=5000000.0, v_leak=0.0, v_th=1.0, v_reset=0.0)
  - Layer 1: LIF (tau_syn_inv=50000.0, tau_mem_inv=5000.0)
- LIF
  - Layer 0: LIF (in_features=1, out_features=40, tau_syn_inv=7142857.0, tau_mem_inv=5000000.0, v_leak=0.0, v_th=1.0, v_reset=0.0)
  - Layer 1: LIF (in_features=40, out_features=28, tau_syn_inv=7142857.0, tau_mem_inv=5000000.0, v_leak=0.0, v_th=1.0, v_reset=0.0)
  - Layer 2: LIF (tau_syn_inv=50000.0, tau_mem_inv=5000.0)
- TC-LIF
  - Preprocessing: Temporal Contrast Step Forward (threshold=0.0961)
  - Layer 0: LIF (in_features=2, out_features=58, tau_syn_inv=7142857.0, tau_mem_inv=5000000.0, v_leak=0.0, v_th=1.0, v_reset=0.0)
  - Layer 1: LIF (tau_syn_inv=50000.0, tau_mem_inv=5000.0)
- RF-CNN
  - Layer 0: R&F (frequencies=range(2, 24, 0.105)MHz, damping_contant=-2.0, threshold=0.4273)
  - Layer 1: Conv2d (in_channels=1, out_channels=4, kernel_size=10, stride=6, padding=0, activation=ReLU)
  - Layer 2: Conv2d (in_channels=4, out_channels=2, kernel_size=5, stride=4, padding=0, activation=ReLU)
  - Layer 3: Linear (input_dim=140, hidden_size=16, activation=ReLU)
  - Layer 4: Linear (input_dim=16, hidden_size=2, activation=ReLU)
- RF-LIF (variable reset, spike functions, recurrency)
  - Layer 0: R&F (frequencies=range(2, 23, 0.105)MHz, damping_contant=-2.0, threshold=0.4273)
  - Layer 1: LIF (in_features=200, out_features=15, tau_syn_inv=7142857.0, tau_mem_inv=5000000.0, v_leak=0.0, v_th=1.0, v_reset=0.0)
  - Layer 2: LIF (tau_syn_inv=50000.0, tau_mem_inv=5000.0)

Bibliography

[1] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network", in *Advances in Neural Information Processing Systems*, vol. 1, Morgan-Kaufmann, 1988.

[2] M. Campbell, A. Hoane, and F.-h. Hsu, "Deep blue", *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[3] "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond", *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 2, pp. 113–120, 2016.

[4] T. S. Ajani, A. L. Imoize, and A. A. Atayero, "An overview of machine learning within embedded and mobile devices–optimizations and applications", *Sensors*, vol. 21, no. 13, p. 4412, 2021.

[5] D. Patterson, J. Gonzalez, Q. Le, *et al.*, *Carbon emissions and large neural network training*, 2021.

[6] L. A. Hart, *How the brain works*. Basic Books, 1975.

[7] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI", *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[8] H. P. Forstner, H. Knapp, H. Jager, *et al.*, "A 77ghz 4-channel automotive radar transceiver in SiGe", in *IEEE Radio Frequency Integrated Circuits Symposium*, IEEE, 2008.

[9] J. R. Speakman, M. E. Anderson, and P. A. Racey, "The energy cost of echolocation in pipistrelle bats (pipistrettus pipistrellus)", *Journal of Comparative Physiology A*, vol. 165, no. 5, pp. 679–685, 1989.

[10] L. Stidsholt, M. Johnson, H. R. Goerlitz, and P. T. Madsen, "Wild bats briefly decouple sound production from wingbeats to increase sensory flow during prey captures", *iScience*, vol. 24, no. 8, p. 102 896, 2021.

[11] X.-S. Yang, "A new metaheuristic bat-inspired algorithm", in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, 2010, pp. 65–74.

[12] E. Covey, J. H. Casseday, *et al.*, "Timing in the auditory system of the bat", *Annual review of physiology*, vol. 61, no. 1, pp. 457–476, 1999.

[13] J. Bechter, C. Sippel, and C. Waldschmidt, "Bats-inspired frequency hopping for mitigation of interference between automotive radars", in *2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, IEEE, 2016, pp. 1–4.

[14] W. Maass, "Networks of spiking neurons: The third generation of neural network models", *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[15] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[16] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems", *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[18] E. T. Rolls and M. J. Tovee, "Processing speed in the cerebral cortex and the neurophysiology of visual masking", *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 257, no. 1348, pp. 9–15, 1994.

[19] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems", *Frontiers in Neuroscience*, vol. 15, 2021.

[20] P. Lennie, "The cost of cortical computation", *Current Biology*, vol. 13, no. 6, pp. 493–497, 2003.

[21] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge Cambridge University Press, 2016.

[22] V. Demin and D. Nekhaev, "Recurrent spiking neural network learning based on a competitive maximization of neuronal activity", *Frontiers in Neuroinformatics*, vol. 12, 2018.

[23] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification", *Frontiers in Neuroscience*, vol. 11, 2017.

[24] X. She and S. Mukhopadhyay, "SPEED: Spiking neural network with event-driven unsupervised learning and near-real-time inference for event-based vision", *IEEE Sensors Journal*, vol. 21, no. 18, pp. 20 578–20 588, 2021.

[25] C. Tan, M. Šarlija, and N. Kasabov, "Spiking neural networks: Background, recent development and the neucube architecture", *Neural Processing Letters*, vol. 52, no. 2, pp. 1675–1701, 2020.

[26] C. D. Schuman, J. P. Mitchell, R. M. Patton, T. E. Potok, and J. S. Plank, "Evolutionary optimization for neuromorphic systems", in *Proceedings of the Neuro-inspired Computational Elements Workshop*, ACM, 2020.

[27] X. Jin, M. Lujan, L. A. Plana, S. Davies, S. Temple, and S. B. Furber, "Modeling spiking neural networks on SpiNNaker", *Computing in science & engineering*, vol. 12, no. 5, pp. 91–97, 2010.

[28] F. Zenke and W. Gerstner, "Limits to high-speed simulations of spiking neural networks using general-purpose computers", *Frontiers in Neuroinformatics*, vol. 8, 2014.

[29] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface", *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[30] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida, "Deep learning in spiking neural networks", *Neural Networks*, vol. 111, pp. 47–63, 2019. arXiv: 1804.08150.

[31] C. Li, R. Chen, C. Moutafis, and S. Furber, "Robustness to noisy synaptic weights in spiking neural networks", in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020.

[32] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification", *Frontiers in Neuroscience*, vol. 12, 2018.

[33] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware", *Frontiers in Neuroscience*, vol. 15, p. 651 141, 2021.

[34] Y. Wang, D. Wu, Y. Wang, *et al.*, "A low-cost hardware-friendly spiking neural network based on binary MRAM synapses, accelerated using in-memory computing", *Electronics*, vol. 10, no. 19, p. 2441, 2021.

[35] H. Shouval, "Spike timing dependent plasticity: A consequence of more fundamental learning rules", *Frontiers in Computational Neuroscience*, 2010.

[36] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems", *Frontiers in Neuroscience*, vol. 7, 2013.

[37] D. A. Moses, S. L. Metzger, J. R. Liu, *et al.*, "Neuroprosthesis for decoding speech in a paralyzed person with anarthria", *New England Journal of Medicine*, vol. 385, no. 3, pp. 217–227, 2021.

[38] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks", *Neural Computation*, vol. 33, no. 4, pp. 899–925, 2021.

[39] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks", *Neural Processing Letters*, 2021.

[40] J. Hille, D. Auge, C. Grassmann, and A. Knoll, "FMCW radar2radar Interference Detection with a Recurrent Neural Network", in *2022 IEEE Radar Conference (RadarConf22)*, New York City, NY, USA: IEEE, 2022, pp. 1–6.

[41] J. Hille, D. Auge, C. Grassmann, and A. Knoll, "Resonate-and-fire neurons for radar interference detection", in *Proceedings of the International Conference on Neuromorphic Systems 2022*, 2022, pp. 1–4.

[42] H. M. Lehmann, J. Hille, C. Grassmann, A. Knoll, and V. Issakov, "Analog spiking neural network based phase detector", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 12, pp. 4837–4846, 2022.

[43] D. Auge, J. Hille, F. Kreutz, E. Mueller, and A. Knoll, "End-to-end spiking neural network for speech recognition using resonating input neurons", in *Artificial Neural Networks and Machine Learning – ICANN 2021*, I. Farkaš, P. Masulli, S. Otte, and S. Wermter, Eds., vol. 12895, Cham: Springer International Publishing, 2021, pp. 245–256.

[44] D. Auge, J. Hille, E. Mueller, and A. Knoll, "Hand gesture recognition in range-doppler images using binary activated spiking neural networks", in *16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, 2021, pp. 01–07.

[45] Bundesministerium für Bildung und Forschung. "Ki-asic - elektronikforschung". German. (2019), [Online]. Available: `https://www.elektronikforschung.de/projekte/ki-asic` (visited on 03/05/2023).

[46] B. Vogginger, F. Kreutz, J. López-Randulfe, *et al.*, "Automotive radar processing with spiking neural networks: Concepts and challenges", *Frontiers in neuroscience*, vol. 16, 2022.

[47] F. Kreutz, D. Scholz, J. Hille, *et al.*, "Continuous inference of time recurrent neural networks for field oriented control", in *IEEE Conference on Artificial Intelligence*, IEEE, 2023.

[48] H. M. Lehmann, J. Hille, C. Grassmann, and V. Issakov, "Spiking Neural Networks based Rate-Coded Logic Gates for Automotive Applications in BiCMOS", in *2021 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, Tel Aviv, Israel: IEEE, 2021, pp. 280–285.

[49] H. M. Lehmann, J. Hille, C. Grassmann, and V. Issakov, "Leaky integrate-and-fire neuron with a refractory period mechanism for invariant spikes", in *2022 17th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*, IEEE, 2022, pp. 365–368.

[50] H. M. Lehmann, J. Hille, C. Grassmann, and V. Issakov, "Direct signal encoding with analog resonate-and-fire neurons", *IEEE Access*, pp. 1–1, 2023.

[51] H. Hertz, *Electric waves: being researches on the propagation of electric action with finite velocity through space*. Dover Publications, 1893.

[52] C. Hülsmeyer, "Verfahren, um entfernte metallische gegenstände mittels elektrischer wellen einem beobachter zu melden", *German Patent*, vol. 165, p. 30, 1904.

[53] G. Capraro, A. Farina, H. Griffiths, and M. Wicks, "Knowledge-based radar signal and data processing: A tutorial review", *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 18–29, 2006.

[54] M. Nosrati and N. Tavassolian, "High-Accuracy Heart Rate Variability Monitoring Using Doppler Radar Based on Gaussian Pulse Train Modeling and FTPR Algorithm", *IEEE Transactions on Microwave Theory and Techniques*, vol. 66, no. 1, pp. 556–567, 2018.

[55] A. Merlo, "Automotive Radar for the Prevention of Collisions", *IEEE Transactions on Industrial Electronics and Control Instrumentation*, vol. IECI-11, no. 1, pp. 1–6, 1964.

[56] C. Waldschmidt, J. Hasch, and W. Menzel, "Automotive Radar — From First Efforts to Future Systems", *IEEE Journal of Microwaves*, vol. 1, no. 1, pp. 135–148, 2021.

[57] M. Weib and J. Ender, "A 3d imaging radar for small unmanned airplanes - artino", in *European Radar Conference, 2005. EURAD 2005.*, 2005, pp. 209–212.

[58] D. Schwarz, N. Riese, I. Dorsch, and C. Waldschmidt, "System Performance of a 79 GHz High-Resolution 4D Imaging MIMO Radar With 1728 Virtual Channels", *IEEE Journal of Microwaves*, vol. 2, no. 4, pp. 637–647, 2022.

[59] BMW AG. "Sensoren in Autos: Sinnesorgane der Assistenzsysteme | BMW.com". de. (), [Online]. Available: `https://www.bmw.com/de/innovation/sensoren-im-auto.html` (visited on 03/21/2023).

[60] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt, "Millimeter-wave technology for automotive radar sensors in the 77 ghz frequency band", *IEEE transactions on microwave theory and techniques*, vol. 60, no. 3, pp. 845–860, 2012.

[61] S. Orru', "Fmcw radar based communication for automotive applications", M.S. thesis, Delft University of Technology, 2020.

[62] S. M. Patole, M. Torlak, D. Wang, and M. Ali, "Automotive radars: A review of signal processing techniques", *IEEE Signal Processing Magazine*, vol. 34, no. 2, pp. 22–35, 2017.

[63] A. G. Stove, "Modern fmcw radar-techniques and applications", in *First European Radar Conference, 2004. EURAD.*, IEEE, 2004, pp. 149–152.

[64] Y. Ju, Y. Jin, and J. Lee, "Design and implementation of a 24 GHz FMCW radar system for automotive applications", in *International Radar Conference*, ISSN: 1097-5764, 2014, pp. 1–4.

[65] T.-Y. J. Kao, Y. Yan, T.-M. Shen, A. Y.-K. Chen, and J. Lin, "Design and Analysis of a 60-GHz CMOS Doppler Micro-Radar System-in-Package for Vital-Sign and Vibration Detection", *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 4, pp. 1649–1659, 2013.

[66] D. E. Barrick, *FM/CW Radar Signals and Digital Processing*. US Department of Commerce, National Oceanic and Atmospheric Administration, Environmental Research Laboratorie, 1973.

[67] G. M. Brooker, "Understanding Millimetre Wave FMCW Radars", *New Zealand*, p. 7, 2005.

[68] P. Rajyalakshmi and G. Raju, "Characteristics of radar cross section with different objects", *International Journal of Electronics and Communication Engineering*, vol. 4, no. 2, pp. 205–216, 2011.

[69] E. F. Knott, J. F. Schaeffer, and M. T. Tulley, *Radar cross section*. SciTech Publishing, 2004.

[70] H. Winner, "Radarsensorik", in *Handbuch Fahrerassistenzsysteme*, H. Winner, S. Hakuli, F. Lotz, and C. Singer, Eds., Wiesbaden: Springer Fachmedien Wiesbaden, 2015, pp. 259–316.

[71] Rayleigh, "XXXI. investigations in optics, with special reference to the spectroscope", *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 49, pp. 261–274, 1879.

[72] J. Fuchs, M. Gardill, M. Lubke, A. Dubey, and F. Lurz, "A machine learning perspective on automotive radar direction of arrival estimation", *IEEE Access*, vol. 10, pp. 6775–6797, 2022.

[73] W.-Q. Wang, "Virtual Antenna Array Analysis for MIMO Synthetic Aperture Radars", en, *International Journal of Antennas and Propagation*, vol. 2012, e587276, 2012.

[74] S. Jayaweera, "Virtual MIMO-based cooperative communication for energy-constrained wireless sensor networks", *IEEE Transactions on Wireless Communications*, vol. 5, no. 5, pp. 984–989, 2006.

[75] Infineon Technologies AG. "Xensiv™ – sensing the world - sensor solutions for automotive, industrial, consumer and iot applications", Infineon Technologies AG. (2022), [Online]. Available: `https://www.infineon.com/dgdl/Infineon-xensiv_sensor_solutions-ProductSelectionGuide-v01_00-EN.pdf?fileId=5546d462636cc8fb0164229c09f51bbe` (visited on 03/15/2023).

[76] A. Palffy, J. Dong, J. F. P. Kooij, and D. M. Gavrila, "CNN Based Road User Detection Using the 3D Radar Cube", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1263–1270, 2020.

[77] S. Alland, W. Stark, M. Ali, and M. Hegde, "Interference in Automotive Radar Systems: Characteristics, Mitigation Techniques, and Current and Future Research", *IEEE Signal Processing Magazine*, vol. 36, no. 5, pp. 45–59, 2019.

[78] D. Lyon, "The discrete fourier transform, part 4: Spectral leakage.", *The Journal of Object Technology*, vol. 8, no. 7, p. 23, 2009.

[79] S. Ivanov, V. Kuptsov, V. Badenko, and A. Fedotov, "An Elaborated Signal Model for Simultaneous Range and Vector Velocity Estimation in FMCW Radar", en, *Sensors*, vol. 20, no. 20, p. 5860, 2020.

[80] M. Smith and P. Varshney, “Intelligent CFAR processor based on data variability”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 837–847, 2000.

[81] C. D. Richmond, “Capon and bartlett beamforming: Threshold effect in direction-of-arrival estimation error and on the probability of resolution”, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, Tech. Rep., 2005.

[82] R. Schmidt, “Multiple emitter location and signal parameter estimation”, *IEEE transactions on antennas and propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[83] R. Roy and T. Kailath, “Esprit-estimation of signal parameters via rotational invariance techniques”, *IEEE Transactions on acoustics, speech, and signal processing*, vol. 37, no. 7, pp. 984–995, 1989.

[84] M. Gall, M. Gardill, T. Horn, and J. Fuchs, “Spectrum-based Single-Snapshot Super-Resolution Direction-of-Arrival Estimation using Deep Learning”, in *German Microwave Conference(GeMiC)*, ISSN: 2167-8022, 2020, pp. 184–187.

[85] A. Alteneiji, U. Ahmad, K. Poon, N. Ali, and N. Almoosa, “Angle of Arrival Estimation in Indoor Environment Using Machine Learning”, in *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, ISSN: 2576-7046, 2021, pp. 1–6.

[86] G. K. Papageorgiou, M. Sellathurai, and Y. C. Eldar, “Deep Networks for Direction-of-Arrival Estimation in Low SNR”, *IEEE Transactions on Signal Processing*, vol. 69, pp. 3714–3729, 2021.

[87] E. Ozanich, P. Gerstoft, and H. Niu, “A feedforward neural network for direction-of-arrival estimation”, *The Journal of the Acoustical Society of America*, vol. 147, no. 3, pp. 2035–2048, 2020.

[88] K. Kaiser, J. Daugalas, J. López-Randulfe, A. Knoll, R. Weigel, and F. Lurz, “Complex-Valued Neural Networks for Millimeter Wave FMCW-Radar Angle Estimations”, in *2022 19th European Radar Conference (EuRAD)*, 2022, pp. 145–148.

[89] G. K. Papageorgiou and M. Sellathurai, “Direction-of-Arrival Estimation in the Low-SNR Regime via a Denoising Autoencoder”, in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, ISSN: 1948-3252, 2020, pp. 1–5.

[90] I. N.K, r. s. a. raja abdullah raja syamsul azmir, and M. I. Saripan, “Artificial Neural Network Approach in Radar Target Classification”, *Journal of Computer Science*, vol. 5, 2009.

[91] F. Kreutz, P. Gerhards, B. Vogginger, K. Knobloch, and C. G. Mayr, “Applied Spiking Neural Networks for Radar-based Gesture Recognition”, in *7th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*, 2021, pp. 1–4.

[92] S. Krebs, B. Duraisamy, and F. Flohr, "A survey on leveraging deep neural networks for object tracking", in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, ISSN: 2153-0017, 2017, pp. 411–418.

[93] M. A. Richards, *Fundamentals of radar signal processing*. McGraw-Hill Education, 2014.

[94] C. N. Hansen, *Understanding active noise cancellation*. CRC Press, 2002.

[95] M. Kunert, "The EU project MOSARIM: A general overview of project objectives and conducted work", in *9th European Radar Conference*, 2012, pp. 1–5.

[96] A. Ossowska, L. Sit, S. Manchala, T. Vogler, K. Krupinski, and U. Luebbert, "IMIKO-Radar Project: Laboratory Interference Measurements of Automotive Radar Sensors", in *2020 21st International Radar Symposium (IRS)*, ISSN: 2155-5753, 2020, pp. 334–338.

[97] E. Yeh, J. Choi, N. Prelcic, C. Bhat, and R. W. Heath Jr, "Security in automotive radar and vehicular networks", *submitted to Microwave Journal*, 2016.

[98] P. Kapoor, A. Vora, and K.-D. Kang, "Detecting and Mitigating Spoofing Attack Against an Automotive Radar", in *IEEE 88th Vehicular Technology Conference (VTC-Fall)*, ISSN: 2577-2465, 2018, pp. 1–6.

[99] kylejbrown17. "Vehicleshapes". (2019), [Online]. Available: `https://github.com/sisl/vehicleshapes` (visited on 03/27/2023).

[100] M. Goppelt, H.-L. Blöcher, and W. Menzel, "Analytical investigation of mutual interference between automotive FMCW radar sensors", in *2011 German Microwave Conference*, ISSN: 2167-8022, 2011, pp. 1–4.

[101] G. M. Brooker, "Mutual Interference of Millimeter-Wave Radar Systems", *IEEE Transactions on Electromagnetic Compatibility*, vol. 49, no. 1, pp. 170–181, 2007.

[102] T. Moon, J. Park, and S. Kim, "BlueFMCW: Random frequency hopping radar for mitigation of interference and spoofing", *EURASIP Journal on Advances in Signal Processing*, vol. 2022, no. 1, p. 4, 2022.

[103] M. Bear, B. Connors, and M. A. Paradiso, *Neuroscience: exploring the brain, enhanced edition: exploring the brain*. Jones & Bartlett Learning, 2020.

[104] B. D. Trapp and G. J. Kidd, "Chapter 1 - Structure of the Myelinated Axon", en, in R. A. Lazzarini, J. W. Griffin, H. Lassman, K.-A. Nave, R. Miller, and B. D. Trapp, Eds., San Diego: Academic Press, 2004, pp. 3–27.

[105] D. Purves, G. J. Augustine, D. Fitzpatrick, W. Hall, A.-S. LaMantia, and L. White, *Neurosciences*. De Boeck Supérieur, 2019.

[106] M. Barnett and P. Larkman, "The action potential", *Practical neurology*, vol. 7, pp. 192–7, 2007.

[107] B. P. Bean, "The action potential in mammalian central neurons", en, *Nature Reviews Neuroscience*, vol. 8, no. 6, pp. 451–465, 2007.

[108] G. G. de Polavieja, A. Harsch, I. Kleppe, H. P. Robinson, and M. Juusola, "Stimulus history reliably shapes action potential waveforms of cortical neurons", *Journal of Neuroscience*, vol. 25, no. 23, pp. 5657–5665, 2005.

[109] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[110] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?", *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[111] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve", *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[112] L. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)", *Brain Research Bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[113] Z. Liu, L. Wang, and S. Xu, "Time-frequency analysis and spiking neural network for motor imagery EEG classification", in *2022 China Automation Congress (CAC)*, ISSN: 2688-0938, 2022, pp. 6458–6462.

[114] J. Huang, B. Vogginger, P. Gerhards, *et al.*, "Real-time Radar Gesture Classification with Spiking Neural Network on SpiNNaker 2 Prototype", in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 362–365.

[115] E. M. Izhikevich, "Resonate-and-fire neurons", *Neural Networks*, vol. 14, no. 6-7, pp. 883–894, 2001.

[116] S. Rotter and M. Diesmann, "Exact digital simulation of time-invariant linear systems with applications to neuronal modeling", *Biological Cybernetics*, vol. 81, no. 5-6, pp. 381–402, 1999.

[117] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[118] G. Orchard, E. P. Frady, D. B. D. Rubin, *et al.*, "Efficient Neuromorphic Signal Processing with Loihi 2", in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, IEEE, 2021, pp. 254–259.

[119] E. P. Frady and F. T. Sommer, "Robust computation with rhythmic spike patterns", *Proceedings of the National Academy of Sciences*, vol. 116, no. 36, pp. 18 050–18 059, 2019.

[120] W. Gerstner, "Chapter 12 A framework for spiking neuron models: The spike response model", en, in *Handbook of Biological Physics*, ser. Neuro-Informatics and Neural Modelling, F. Moss and S. Gielen, Eds., vol. 4, North-Holland, 2001, pp. 469–516.

[121] A. Galves and E. Löcherbach, "Infinite Systems of Interacting Chains with Memory of Variable Length—A Stochastic Model for Biological Neural Nets", en, *Journal of Statistical Physics*, vol. 151, no. 5, pp. 896–921, 2013.

[122] R. FitzHugh, "Mathematical models of threshold phenomena in the nerve membrane", en, *The bulletin of mathematical biophysics*, vol. 17, no. 4, pp. 257–278, 1955.

[123] S. W. Harden. "pyHH". original-date: 2019-10-19T20:14:30Z. (2023), [Online]. Available: `https://github.com/swharden/pyHH/blob/master/src/pyhh/models.py` (visited on 03/30/2023).

[124] C. Graßmann, "Simulation pulsverarbeitender neuronaler Netze", deu, Ph.D. dissertation, Universitäts- und Landesbibliothek Bonn, 2003.

[125] L. Mo and Z. Tao, "EvtSNN: Event-driven SNN simulator optimized by population and pre-filtering", *Frontiers in Neuroscience*, vol. 16, 2022.

[126] B. Zuber, I. Nikonenko, P. Klauser, D. Muller, and J. Dubochet, "The mammalian central nervous synaptic cleft contains a high density of periodically organized complexes", *Proceedings of the National Academy of Sciences*, vol. 102, no. 52, pp. 19 192–19 197, 2005.

[127] W. Volknandt, "The synaptic vesicle and its targets", en, *Neuroscience*, vol. 64, no. 2, pp. 277–300, 1995.

[128] A. Macdermott, L. Role, and S. Siegelbaum, "MacDermott AB, Role LW, Siegelbaum SA. Presynaptic ionotropic receptors and the control of transmitter release. Annu Rev Neurosci 22: 443-485", *Annual review of neuroscience*, vol. 22, pp. 443–85, 1999.

[129] J. .-. Pin and R. Duvoisin, "The metabotropic glutamate receptors: Structure and functions", en, *Neuropharmacology*, vol. 34, no. 1, pp. 1–26, 1995.

[130] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, S. Mack, *et al.*, *Principles of Neural Science*. McGraw-hill New York, 2000, vol. 4.

[131] B. Katz and R. Miledi, "The Measurement of Synaptic Delay, and the Time Course of Acetylcholine Release at the Neuromuscular Junction", *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 161, no. 985, pp. 483–495, 1965.

[132] S. Chung, X. Li, and S. B. Nelson, "Short-Term Depression at Thalamocortical Synapses Contributes to Rapid Adaptation of Cortical Sensory Responses In Vivo", en, *Neuron*, vol. 34, no. 3, pp. 437–446, 2002.

[133] J. D. Goutman and E. Glowatzki, "Short-term facilitation modulates size and timing of the synaptic response at the inner hair cell ribbon synapse", *The Journal of Neuroscience*, vol. 31, no. 22, pp. 7974–7981, 2011.

[134] T. V. P. Bliss and S. F. Cooke, "Long-term potentiation and long-term depression: A clinical perspective", en, *Clinics*, vol. 66, pp. 3–17, 2011.

[135] L. F. Abbott and W. G. Regehr, “Synaptic computation”, en, *Nature*, vol. 431, no. 7010, pp. 796–803, 2004.

[136] M. Tsodyks, K. Pawelzik, and H. Markram, “Neural networks with dynamic synapses”, *Neural Computation*, vol. 10, no. 4, pp. 821–835, 1998.

[137] W. Maass and H. Markram, “Synapses as dynamic memory buffers”, en, *Neural Networks*, vol. 15, no. 2, pp. 155–161, 2002.

[138] L. Xiao, D.-K. Zhang, Y.-q. Li, P.-J. Liang, and S. Wu, “Adaptive neural information processing with dynamical electrical synapses”, *Frontiers in Computational Neuroscience*, vol. 7, 2013.

[139] C. Lee, G. Srinivasan, P. Panda, and K. Roy, “Deep Spiking Convolutional Neural Network Trained With Unsupervised Spike-Timing-Dependent Plasticity”, *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 384–394, 2019.

[140] K. Nakajima, “Physical reservoir computing—an introductory perspective”, en, *Japanese Journal of Applied Physics*, vol. 59, no. 6, p. 060 501, 2020.

[141] H. Jaeger, “The “echo state” approach to analysing and training recurrent neural networks – with an Erratum note”, *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[142] W. Maass, T. Natschläger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations”, *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[143] N. Kasabov, N. M. Scott, E. Tu, *et al.*, “Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications”, en, *Neural Networks*, Special Issue on ”Neural Network Learning in Big Data”, vol. 78, pp. 1–14, 2016.

[144] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, “Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware”, in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, 2016, pp. 1–8.

[145] B. Rueckauer and S.-C. Liu, “Conversion of analog to spiking neural networks using sparse temporal coding”, in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2379-447X, 2018, pp. 1–5.

[146] D. Pérez-González and M. S. Malmierca, “Adaptation in the auditory system: An overview”, *Frontiers in Integrative Neuroscience*, vol. 8, p. 19, 2014.

[147] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

[148] S. Löwel and W. Singer, “Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity”, *Science*, vol. 255, no. 5041, pp. 209–212, 1992.

[149] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity", en, *Nature Neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[150] Y. Hao, X. Huang, M. Dong, and B. Xu, "A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule", en, *Neural Networks*, vol. 121, pp. 387–395, 2020.

[151] T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.

[152] S. Schliebs and N. Kasabov, "Evolving spiking neural network—a survey", *Evolving Systems*, vol. 4, no. 2, pp. 87–98, 2013.

[153] S.-i. Amari, "Backpropagation and stochastic gradient descent method", en, *Neurocomputing*, vol. 5, no. 4, pp. 185–196, 1993.

[154] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks", *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[155] P. Werbos, "Backpropagation through time: What it does and how to do it", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[156] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model", en, *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.

[157] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", en, *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[158] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[159] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain", en, *Current Opinion in Neurobiology*, Machine Learning, Big Data, and Neuroscience, vol. 55, pp. 82–89, 2019.

[160] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time", *Advances in neural information processing systems*, vol. 31, 2018.

[161] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers", *Advances in neural information processing systems*, vol. 31, 2018.

[162] F. Naveros, J. A. Garrido, R. R. Carrillo, E. Ros, and N. R. Luque, "Event-and time-driven techniques using parallel cpu-gpu co-processing for spiking neural networks", *Frontiers in neuroinformatics*, vol. 11, p. 7, 2017.

[163] Y. Yan, H. Chu, Y. Jin, Y. Huan, Z. Zou, and L. Zheng, "Backpropagation with sparsity regularization for spiking neural network learning", *Frontiers in Neuroscience*, vol. 16, 2022.

[164] D. Salaj, A. Subramoney, C. Kraisnikovic, G. Bellec, R. Legenstein, and W. Maass, "Spike frequency adaptation supports network computations on temporally dispersed information", *Elife*, vol. 10, e65459, 2021.

[165] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks", *arXiv preprint arXiv:1803.03635*, 2018.

[166] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks", *arXiv preprint arXiv:1711.05136*, 2017.

[167] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?", *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.

[168] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies", *Science*, vol. 319, no. 5866, pp. 1108–1111, 2008.

[169] E. D. Adrian and Y. Zotterman, "The impulses produced by sensory nerve endings", *The Journal of Physiology*, vol. 61, no. 4, pp. 465–483, 1926.

[170] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm", in *2011 IEEE custom integrated circuits conference (CICC)*, IEEE, 2011, pp. 1–4.

[171] S. R. Kheradpisheh and T. Masquelier, "Temporal backpropagation for spiking neural networks with one spike per neuron", *International Journal of Neural Systems*, vol. 30, no. 06, p. 2 050 027, 2020.

[172] F. Theunissen and J. P. Miller, "Temporal encoding in nervous systems: A rigorous definition", *Journal of computational neuroscience*, vol. 2, no. 2, pp. 149–162, 1995.

[173] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2003.

[174] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 1311–1318, 2019.

[175] M. Arsalan, A. Santra, and V. Issakov, "RadarSNN: A resource efficient gesture sensing system based on mm-wave radar", *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 4, pp. 2451–2461, 2022.

[176] F. Faghihi, S. Cai, and A. A. Moustafa, "A neuroscience-inspired spiking neural network for EEG-based auditory spatial attention detection", *Neural Networks*, vol. 152, pp. 555–565, 2022.

[177] S. J. Thorpe, "Spike arrival times: A highly efficient coding scheme for neural networks", *Parallel processing in neural systems*, pp. 91–94, 1990.

[178] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system", en, *Nature*, vol. 381, no. 6582, pp. 520–522, 1996.

[179] T. J. Gawne, T. W. Kjaer, and B. J. Richmond, "Latency: Another potential code for feature binding in striate cortex", *Journal of neurophysiology*, vol. 76, no. 2, pp. 1356–1360, 1996.

[180] M. A. Montemurro, M. J. Rasch, Y. Murayama, N. K. Logothetis, and S. Panzeri, "Phase-of-firing coding of natural visual stimuli in primary visual cortex", *Current Biology*, vol. 18, no. 5, pp. 375–380, 2008.

[181] G. Portelli, J. M. Barrett, G. Hilgen, *et al.*, "Rank order coding: A retinal information decoding strategy revealed by large-scale multielectrode array retinal recordings", *eneuro*, vol. 3, no. 3, ENEURO.0134–15.2016, 2016.

[182] R. Galambos and H. Davis, "The response of single auditory-nerve fibers to acoustic stimulation", *Journal of Neurophysiology*, vol. 6, no. 1, pp. 39–57, 1943.

[183] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events", en, *Nature Neuroscience*, vol. 7, no. 2, pp. 170–177, 2004.

[184] T. W. Margrie and A. T. Schaefer, "Theta oscillation coupled spike latencies yield computational vigour in a mammalian sensory system", en, *The Journal of Physiology*, vol. 546, no. 2, pp. 363–374, 2003.

[185] N. M. Abraham, H. Spors, A. Carleton, T. W. Margrie, T. Kuner, and A. T. Schaefer, "Maintaining Accuracy at the Expense of Speed: Stimulus Similarity Defines Odor Discrimination Time in Mice", en, *Neuron*, vol. 44, no. 5, pp. 865–876, 2004.

[186] R. Batllori, C. B. Laramee, W. Land, and J. D. Schaffer, "Evolving spiking neural networks for robot control", en, *Procedia Computer Science*, Complex adaptive sysytems, vol. 6, pp. 329–334, 2011.

[187] A. Sboev, D. Vlasov, R. Rybka, and A. Serenko, "Spiking neural network reinforcement learning method based on temporal coding and STDP", en, *Procedia Computer Science*, Postproceedings of the 9th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2018 (Ninth Annual Meeting of the BICA Society), held August 22-24, 2018 in Prague, Czech Republic, vol. 145, pp. 458–463, 2018.

[188] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation", en, *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.

[189] C. Kayser, M. A. Montemurro, N. K. Logothetis, and S. Panzeri, "Spike-Phase Coding Boosts and Stabilizes Information Carried by Spatial and Temporal Spike Patterns", en, *Neuron*, vol. 61, no. 4, pp. 597–608, 2009.

[190] C. M. Gray, P. König, A. K. Engel, and W. Singer, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties", en, *Nature*, vol. 338, no. 6213, pp. 334–337, 1989.

[191] S. Thorpe and J. Gautrais, "Rank order coding", *Computational Neuroscience: Trends in Research, 1998*, pp. 113–118, 1998.

[192] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 s Latency Asynchronous Temporal Contrast Vision Sensor", *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[193] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system", in *2007 IEEE International Symposium on Circuits and Systems*, ISSN: 2158-1525, 2007, pp. 845–848.

[194] B. Petro, N. Kasabov, and R. M. Kiss, "Selection and Optimization of Temporal Spike Encoding Methods for Spiking Neural Networks", *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.

[195] D. Auge and E. Mueller, "Resonate-and-Fire Neurons as Frequency Selective Input Encoders for Spiking Neural Networks", p. 8, 2020.

[196] A. Mertins and D. A. Mertins, *Signal analysis: wavelets, filter banks, time-frequency transforms and applications*. John Wiley & Sons, Inc., 1999.

[197] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform", *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 2, pp. 236–243, 1984.

[198] M. A. De Jesus, M. Teixeira, L. Vicente, and Y. Rodriguez, "Nonuniform discrete short-time fourier transform a goertzel filter bank versus a fir filtering approach", in *2006 49th IEEE International Midwest Symposium on Circuits and Systems*, IEEE, vol. 2, 2006, pp. 188–192.

[199] S. Bohte, H. La Poutre, and J. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks", *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 426–435, 2002.

[200] G. V. Joseph and V. Pakrashi, "Spiking Neural Networks for Structural Health Monitoring", en, *Sensors*, vol. 22, no. 23, p. 9245, 2022.

[201] M. S. Lewicki, "Efficient coding of natural sounds", en, *Nature Neuroscience*, vol. 5, no. 4, pp. 356–363, 2002.

[202] C. C. Aggarwal, *Outlier Analysis*. Cham: Springer International Publishing, 2017.

[203] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[204] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements", in *International Workshop on Quality of Service*, Springer, 2003, pp. 381–398.

[205] F. Laghezza, F. Jansen, and J. Overdevest, "Enhanced interference detection method in automotive fmcw radar systems", in *2019 20th International Radar Symposium (IRS)*, IEEE, 2019, pp. 1–7.

[206] J. Wang, "Cfar-based interference mitigation for fmcw automotive radar systems", *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[207] J.-H. Choi, H.-B. Lee, J.-W. Choi, and S.-C. Kim, "Mutual interference suppression using clipping and weighted-envelope normalization for automotive fmcw radar systems", *IEICE Transactions on Communications*, vol. 99, no. 1, pp. 280–287, 2016.

[208] M. Umehira, T. Okuda, X. Wang, S. Takeda, and H. Kuroda, "An adaptive interference detection and suppression scheme using iterative processing for automotive fmcw radars", in *2020 IEEE Radar Conference (RadarConf20)*, IEEE, 2020, pp. 1–5.

[209] Y. Watanabe and K. Natsume, *Interference determination method and fmcw radar using the same*, US Patent 7,187,321, 2007.

[210] F. Jin and S. Cao, "Automotive radar interference mitigation using adaptive noise canceller", *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3747–3754, 2019.

[211] J. Mun, H. Kim, and J. Lee, "A Deep Learning Approach for Automotive Radar Interference Mitigation", in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, USA: IEEE, 2018, pp. 1–5.

[212] J. Fuchs, A. Dubey, M. Lubke, R. Weigel, and F. Lurz, "Automotive radar interference mitigation using a convolutional autoencoder", in *2020 IEEE International Radar Conference (RADAR)*, Washington, DC, USA: IEEE, 2020, pp. 315–320.

[213] J. Kim, S. Lee, Y.-H. Kim, and S.-C. Kim, "Classification of Interference Signal for Automotive Radar Systems With Convolutional Neural Network", *IEEE Access*, vol. 8, pp. 176 717–176 727, 2020.

[214] R. Zhang and S. Cao, "Support vector machines for classification of automotive radar interference", in *2018 IEEE Radar Conference (RadarConf18)*, ISSN: 2375-5318, 2018, pp. 0366–0371.

[215] B. R. Kiran, D. M. Thomas, and R. Parakkal, "An Overview of Deep Learning Based Methods for Unsupervised and Semi-Supervised Anomaly Detection in Videos", en, *Journal of Imaging*, vol. 4, no. 2, p. 36, 2018.

[216] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, vol. 27, 2014.

[217] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems", *Advances in neural information processing systems*, vol. 9, 1996.

[218] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition", *arXiv preprint arXiv:1402.1128*, 2014.

[219] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures", *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[220] N. Nuntalid, K. Dhoble, and N. Kasabov, "Eeg classification with bsa spike encoding algorithm and evolving probabilistic spiking neural network", in *International conference on neural information processing*, Springer, 2011, pp. 451–460.

[221] L. Mosley, "A balanced approach to the multi-class imbalance problem", *Doctor of Philosophy Thesis, Iowa State University of Science and Technology, USA*, 2013.

[222] J. Han, M. Kamber, and J. Pei, "8 - Classification: Basic Concepts", en, in ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Boston: Morgan Kaufmann, 2012, pp. 327–391.

[223] C. Pehle and J. E. Pedersen, *Norse - A deep learning library for spiking neural networks*, version 0.0.7, Documentation: https://norse.ai/docs/, 2021.

[224] J. Rock, M. Toth, P. Meissner, and F. Pernkopf, "Deep Interference Mitigation and Denoising of Real-World FMCW Radar Signals", in *2020 IEEE International Radar Conference (RADAR)*, IEEE, 2020, pp. 624–629.

[225] D. B. Rubin, "Statistical disclosure limitation", *Journal of official Statistics*, vol. 9, no. 2, pp. 461–468, 1993.

[226] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data", *Journal of artificial intelligence research*, vol. 11, pp. 131–167, 1999.

[227] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey", *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[228] A. Torralba and A. A. Efros, "Unbiased look at dataset bias", in *CVPR 2011*, IEEE, 2011, pp. 1521–1528.

[229] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias", in *European Conference on Computer Vision*, Springer, 2012, pp. 158–171.

[230] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning", *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[231] V. V. Danilov, D. Y. Kolpashchikov, O. M. Gerget, *et al.*, "Use of semi-synthetic data for catheter segmentation improvement", en, *Computerized Medical Imaging and Graphics*, vol. 106, p. 102 188, 2023.

[232] D. Scharf, S. Saadani, M. Schneider, and A. Hoss, "A semi-automated multi-sensor data labeling process for deep learning in automotive radar environment", in *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, IEEE, 2022.

[233] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", *Advances in neural information processing systems*, vol. 32, 2019.

[234] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework", in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[235] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization", *Advances in neural information processing systems*, vol. 24, 2011.

[236] W. Rall, "Membrane potential transients and membrane time constant of motoneurons", *Experimental neurology*, vol. 2, no. 5, pp. 503–532, 1960.

[237] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance", in *International conference on machine learning*, PMLR, 2014, pp. 754–762.

[238] R. Rubin, L. F. Abbott, and H. Sompolinsky, "Balanced excitation and inhibition are required for high-capacity, noise-robust neuronal selectivity", *Proceedings of the National Academy of Sciences*, vol. 114, no. 44, 2017.

[239] V. S. Sohal and J. L. Rubenstein, "Excitation-inhibition balance as a framework for investigating mechanisms in neuropsychiatric disorders", *Molecular psychiatry*, vol. 24, no. 9, pp. 1248–1257, 2019.

[240] X. Ying, "An overview of overfitting and its solutions", in *Journal of physics: Conference series*, IOP Publishing, vol. 1168, 2019, p. 022 022.

[241] X. Liu, M. Yan, L. Deng, *et al.*, "General spiking neural network framework for learning trajectory from noisy mmwave radar", *Neuromorphic Computing and Engineering*, 2022.

[242] S. Kuo, B. Lee, and W. Tian, *Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications*. Wiley, 2013.

[243] H. C. Wee, S. Watson, R. Patz, H. Griffiths, and R. J. Williams, "A magnetic induction tomography system with sub-millidegree phase noise and high long-term phase stability", in *IFMBE Proceedings*, Springer Berlin Heidelberg, 2009, pp. 744–747.

[244] J. Guo and C. Yang, "Highly stabilized phase-shifted fiber bragg grating sensing system for ultrasonic detection", *IEEE Photonics Technology Letters*, vol. 27, no. 8, pp. 848–851, 2015.

[245] H. Klingbeil, "A fast DSP-based phase-detector for closed-loop RF control in synchrotrons", *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1209–1213, 2005.

[246] H. Yan, X. Liao, and D. Hua, "The phase sensitivity and response time of an x-band dual channel microwave phase detector", in *2015 IEEE SENSORS*, IEEE, 2015.

[247] S. R. Kurtz, "Mixers as phase detectors", *Watkins-Johnson Tech-Notes*, vol. 5, no. 1, pp. 1–8, 1973.

[248] P. König, A. K. Engel, and W. Singer, "Integrator or coincidence detector? the role of the cortical neuron revisited", *Trends in Neurosciences*, vol. 19, no. 4, pp. 130–137, 1996.

[249] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)", *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[250] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit", in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03.*, IEEE, 2003.

[251] J. Benda and A. V. M. Herz, "A universal model for spike-frequency adaptation", *Neural Computation*, vol. 15, no. 11, pp. 2523–2564, 2003.

[252] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI", *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, 2007.

[253] B. Wang, W. Ke, J. Guang, *et al.*, "Firing Frequency Maxima of Fast-Spiking Neurons in Human, Monkey, and Mouse Neocortex", *Frontiers in Cellular Neuroscience*, vol. 10, 2016.

[254] E. Ahissar, "Temporal-code to rate-code conversion by neuronal phase-locked loops", *Neural Computation*, vol. 10, no. 3, pp. 597–650, 1998.

[255] M. Lees, "Digital beamforming calibration for FMCW radar", 2, vol. 25, IEEE, 1989, pp. 281–284.

[256] M. Nalezinski and C. Seisenberger, "Radar arrangement", US7990313B2, 2007.

[257] D.-B. Radars and for Automotive, "Analysis of amplitude and phase errors in digital-beamforming radars for automotive applications", in *2020 21st International Radar Symposium (IRS)*, Warsaw University of Technology, 2020.

[258] C. Vasanelli, F. Roos, A. Durr, *et al.*, "Calibration and direction-of-arrival estimation of millimeter-wave radars: A practical introduction", *IEEE Antennas and Propagation Magazine*, vol. 62, no. 6, pp. 34–45, 2020.

[259] R. Kreiser, A. Renner, V. R. C. Leite, *et al.*, "An on-chip spiking neural network for estimation of the head pose of the icub robot", *Frontiers in Neuroscience*, vol. 14, 2020.

[260] A. Shalumov, R. Halaly, and E. E. Tsur, "LiDAR-driven spiking neural network for collision avoidance in autonomous driving", *Bioinspiration & Biomimetics*, vol. 16, no. 6, p. 066 016, 2021.

[261] J. Cong, X. Wang, M. Huang, and L. Wan, "Robust DOA estimation method for MIMO radar via deep neural networks", *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7498–7507, 2021.

[262] D. Gabor, "Theory of communication. part 1: The analysis of information", *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[263] L. Li and A. M. Wyrwicz, "Parallel 2d FFT implementation on FPGA suitable for real-time MR image processing", *Review of Scientific Instruments*, vol. 89, no. 9, p. 093 706, 2018.

[264] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for energy-efficient object detection", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 11 270–11 277, 2020.

[265] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding", in *Computer Vision ECCV 2020*, Springer International Publishing, 2020, pp. 388–404.

[266] G. E. Hinton, "The forward-forward algorithm: Some preliminary investigations", *CoRR*, vol. abs/2212.13345, 2022. arXiv: 2212.13345.

[267] K. Friston and S. Kiebel, "Predictive coding under the free-energy principle", *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1521, pp. 1211–1221, 2009.