UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING

MASTER'S DEGREE IN ICT FOR INTERNET AND MULTIMEDIA

"TITLE: Spiking Neural Network-Based Target Detection and Ranging for Integrated Sensing and Communication."

Supervisor: Prof. / Dott. Jacopo Pegoraro

Candidate: Shahla Sadeghzadehdarandash

Co-supervisor: Prof. / Dott……

ACADEMIC YEAR 2024 – 2025

Graduation date 12/09/2025

Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for internet and multimedia

Spiking Neural Network-Based Target Detection and Ranging for Integrated Sensing and Communication

Supervisor
Dr. Jacopo Pegoraro

Master Candidate
Shahla Sadeghzadehdarandash

Academic year
2024/2025

To my family and friends.

Abstract

Integrated Sensing and Communication (ISAC) is an emerging field that brings together two technologies that we often think of as separate: sensing and wireless data transfer. In a traditional system, there is one device or signal for 'seeing' objects (like a radar) and another for 'communicating' with other devices. ISAC combines these into a single system, letting us use the same signals, antennas, and hardware to both sense the environment and transfer data. This saves spectrum, power, and cost, and it is exactly what future applications like smart cars, drones, and the Internet of Things will need. In our work, we use a special kind of neural network called a Spiking Neural Network (SNN). Unlike ordinary deep networks, SNNs communicate via short electrical "spikes," much like real neurons in the brain. We feed these SNNs with pulse-position–modulated impulse radio signals. The network has two 'heads': one head learns to decide whether a target is present (classification), and the other predicts the target's propagation delay (regression), all at once. To test our system under realistic conditions, we built simulated wireless channels, including random background echoes (clutter) and noise. By changing the statistical properties of the target and clutter signals, we created 12 datasets to see how well the network could learn to separate "real" targets from distracting echoes. This approach explores how statistical separation affects target detectability via SNNs.

Sommario

La Rilevazione e Comunicazione Integrata (ISAC) è un campo nuovo che unisce due tecnologie che di solito consideriamo separate: il rilevamento (sensing) e il trasferimento di dati wireless. In un sistema tradizionale, c'è un dispositivo o un segnale per "vedere" gli oggetti (come un radar) e un altro per "comunicare" con altri dispositivi. ISAC riunisce questi due compiti in un unico sistema, permettendo di usare gli stessi segnali, antenne e hardware sia per rilevare l'ambiente sia per inviare dati. Questo risparmia spettro, energia e costi ed è proprio ciò che servirà a future applicazioni come auto intelligenti, droni e Internet delle Cose. Nel nostro lavoro usiamo un tipo speciale di rete neurale chiamata Rete Neurale a Spike (SNN). A differenza delle reti profonde tradizionali, le SNN comunicano tramite brevi "spike" elettrici, proprio come i neuroni reali del cervello. Alleniamo queste SNN con segnali radio a impulso modulati nella posizione dei picchi (pulse-position modulation). La rete ha due "teste": una testa impara a capire se c'è un bersaglio (classificazione) e l'altra predice il ritardo del segnale di ritorno del bersaglio (regressione), tutto insieme. Per testare il nostro sistema in condizioni realistiche, abbiamo creato canali wireless simulati che includono echi di fondo casuali (clutter) e rumore. Cambiando le caratteristiche statistiche dei segnali del bersaglio e del clutter, abbiamo generato 12 set di dati per vedere quanto bene la rete riesce a separare i "veri" bersagli dagli echi di disturbo. Questo approccio mostra come la differenza tra i dati (separazione statistica) influenzi la capacità delle SNN di individuare il bersaglio.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ISAC . . . . . Integrated Sensing And Communication

RADAR . . . . . Radio Detection and Ranging

DFRC . . . . . Dual-Functional Radar and Communication

BS . . . . . Base Station

DNNs . . . . . Deep Neural Networks

SNNs . . . . . Spiking Neural Networks

CNNs . . . . . Convolutional Neural Networks

RNNs . . . . . Recurrent Neural Networks

IR . . . . . Impulse Radio

N-ISAC . . . . . Neuromorphic Integrated Sensing And Communication

SNR . . . . . Signal to Noise Ratio

SPK . . . . . spike

JCAS . . . . . Joint Communication And Sensing

RCS . . . . . Radar Cross Section

AWGN . . . . . Additive White Gaussian Noise

UWB . . . . . Ultra-Wideband

PPM . . . . . Pulse Position Modulation

ANNs . . . . . artificial neural networks

LIF . . . . . Leaky Integrate-and-Fire

IF . . . . . Integrate-and-Fire ()

CuBa . . . . . Current-Based ()

SRM . . . . . SpikeResponseModel()

GLM . . . . . Generalized Linear Model

BPTT . . . . . Backpropagation Through Time

MSE . . . . . Mean Squared Error

CDF . . . . . Cumulative Distribution Function

TP . . . . . TruePositives()

TN . . . . . TrueNegatives()

FP . . . . . FalsePositives()

FN . . . . . FalseNegatives()

ROC . . . . . Receiver Operating Characteristic Curve

AUC . . . . . Area Under the Curve

1 Introduction

1.1 Integrated Sensing and Communication (ISAC)

For many years, wireless communication and radar sensing systems have developed separately. Each field has made great progress using its own technologies and methods. Communication systems have focused on transmitting data efficiently, while radar systems have focused on detecting and interpreting signals from the environment [2, 3]. However, as technology evolves, especially with the spread of smart environments and autonomous devices, the line between sensing and communication is becoming blurred [4, 5].

A new method called Integrated Sensing and Communication (ISAC) has been developed to meet the growing need for combining sensing and communication in modern wireless systems. ISAC aims to unify sensing and communication tasks by using the same resources such as a common frequency band, waveform, and hardware, so both functions can operate at the same time without requiring extra equipment or systems [4, 6, 7]. Shifting from separate to integrated systems leads to better use of spectrum, lower energy consumption, and reduced delays. These three improvements are essential for real-time applications at the edge of networks (the part of the network near end devices like sensors or vehicles, where fast data processing and low delay are needed) [5, 8]. Figure 1.1 shows a real system where radar and communication are combined, allowing the base station to support both radar sensing and wireless communication tasks at the same time.

This integration is becoming essential in real-world applications. For instance, autonomous

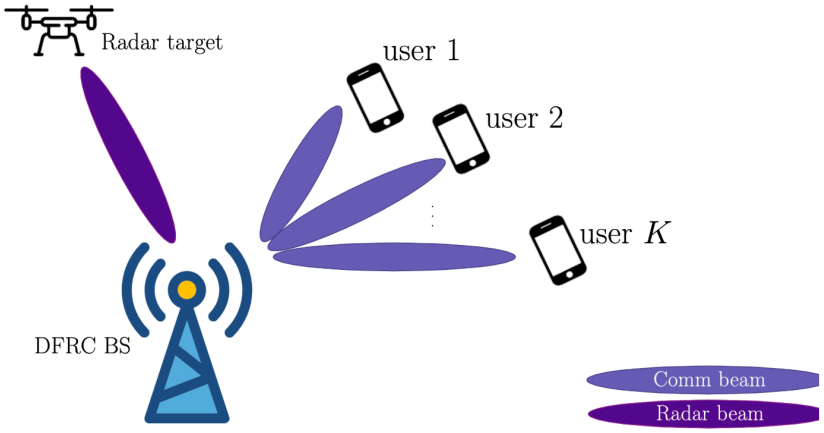


Figure 1.1: An illustration of a practical DFRC system, where a base station simultaneously transmits communication beams to multiple users and radar beams to detect environmental targets, from [9].

vehicles must sense their surroundings while communicating with other vehicles and infrastructure [10]. IoT devices in smart cities and industries also need to collect environmental data and stay connected to the cloud [11]. As 6G networks emerge, demands for lower latency, more connections, and better sensing will increase [12]. Such demands highlight the need for systems that do more with less spectrum, power, and hardware.

Despite the potential of ISAC, building efficient and practical systems remains a major challenge. Many current ISAC solutions use traditional signal processing methods or deep learning models for tasks such as target detection, range estimation, and object classification [6, 7]. Although deep neural networks (DNNs) perform well in these areas, they have serious limitations. They require high amounts of computation and memory, which makes them unsuitable for edge devices that must work in real time with limited power [13].

To address these limitations, researchers are exploring more energy-efficient alternatives that still offer high performance. One promising approach is neuromorphic computing, especially the use of Spiking Neural Networks (SNNs) [14, 15]. SNNs work like biological neurons by processing information through sparse, event-driven spike signals [13, 8]. This method allows low-latency, asynchronous computing and greatly reduces energy consumption [16]. These features make SNNs a good fit for ISAC applications, where real-time decisions must be made with limited power.

In this context, the challenge is how to effectively combine SNNs models with radar com-

munication systems to support both accurate target detection and range estimation. At the same time, any solution must be robust against complex channel conditions, such as multipath propagation, noise, and clutter, which often occur in real-world scenarios. This challenge forms the main motivation for the work presented in this thesis.

1.2 Related Work

The integration of sensing and communication systems, now commonly known as ISAC, has attracted significant research interest in recent years. Early ISAC methods mainly used classical signal processing techniques, such as matched filtering, delay estimation, and Doppler analysis. These approaches allowed communication and radar tasks to run separately but often needed different resources, which led to redundancy and inefficient use of the spectrum [6].

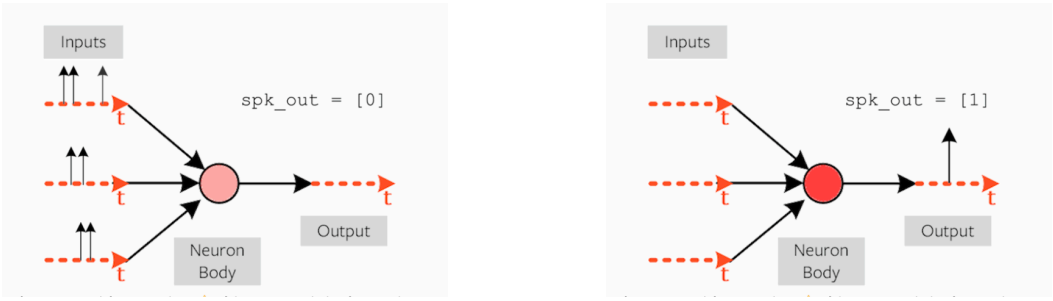
To overcome these limitations, recent research has explored deep learning models for ISAC tasks. In particular, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been used successfully for tasks such as range estimation, moving target classification, and clutter suppression [3, 17]. These models automatically learn complex spatial and temporal features from raw data, providing a powerful alternative to traditional hand-crafted methods. However, their high computational and energy costs limit their use in real time, embedded ISAC applications.

Recognizing the need for more energy-efficient solutions, researchers have turned their attention to neuromorphic computing, especially SNNs. SNNs are inspired by biological neurons, which communicate via discrete events called spikes. This type of processing supports asynchronous and time-efficient computation, which is particularly useful in sensor-rich and time-sensitive environments. [13]. The behavior of a spiking neuron, producing an output spike only when receiving sufficient stimulation, is shown in Figure 1.2. Training SNNs for complex tasks has become more practical with the introduction of surrogate gradient methods, which allow spiking networks to achieve performance levels comparable to traditional deep neural networks [14]. The idea of using SNNs within ISAC frameworks began to emerge around 2021, with early works such as Shi et al. [5] which proposing a neuromorphic ISAC system that uses SNNs for both sensing and semantic communication tasks. Their work demonstrated that SNNs can effectively extract important features from radar echoes while transmitting compressed information at the semantic level, greatly reducing communication overhead. Similarly, Chen et al. [8] showed that SNNs can efficiently

process wireless signals using impulse radio (IR) transmissions, which are naturally sparse in the time domain and well suited to event-driven architectures.

While these studies show the potential of combining SNNs and ISAC, several challenges remain. Many existing works focus on improving either sensing or communication, but not both together in a fully integrated system. In addition, the robustness of SNN-based ISAC systems under real-world conditions—such as multipath propagation, noise, and clutter—has not been fully investigated. Most current models also lack detailed experimental validation under different physical conditions, making it difficult to evaluate their practical performance in real environments.

This thesis addresses these gaps by proposing a dual-head SNNs architecture that jointly performs target detection (classification) and time-of-propagation estimation (regression) for ISAC applications. The study leverages the fact that, in the N-ISAC system, the only statistical difference between the radar target and the clutter lies in the distribution of their complex channel gains (β_0 for the target and β_c for clutter). The main goal is to explore how this statistical separability affects the network's ability to detect the presence of a target. By systematically varying the channel conditions in simulated datasets—including multipath fading, clutter strength, and SNR—the experiments analyze the impact of this separation on the performance of the SNNs. This approach helps to understand how well SNNs can use small statistical differences to detect targets in real-time N-ISAC systems.



(a) Insufficient stimulation ($spk\_out = 0$)

(b) Sufficient stimulation ($spk\_out = 1$)

Figure 1.2: Illustration of spiking neuron behavior. (a) The neuron receives insufficient input and remains silent ($spk\_out = 0$). (b) The neuron fires an output spike ($spk\_out = 1$) once the incoming inputs surpass the firing threshold, from [18].

2 Background

2.1 Joint Communication and Sensing

As wireless systems advance rapidly, the separation between communication and sensing is becoming less clear. This chapter introduces the key concepts of Joint Communication and Sensing (JCS), highlighting its role and integration in radar-based systems.

2.1.1 Fundamentals of Radar and Sensing

Radar systems have played a main role in detecting, locating, and tracking objects in a wide range of applications, from aviation and military surveillance to autonomous vehicles. The basic principle of radar involves transmitting an electromagnetic signal toward a target and analyzing the reflected signal, or echo, that returns to the receiver. By measuring properties such as time delay, frequency shift, and signal strength of the received echo, radar systems can estimate important parameters like the target's range, speed, and angular position [19].

At the core of radar sensing is the concept of time delay. Since electromagnetic waves travel at the speed of light, the time it takes for a transmitted signal to reach a target and return is directly related to the distance between the radar and the object. If τ denotes the round-trip time delay, the range R to the target is given by

$$R = \frac{c\tau}{2}, \tag{2.1}$$

where c is the speed of light in free space. Accurate measurement of τ enables the radar system to determine the location of objects with high precision.

In practice, radar systems must deal with various challenges, notably textitclutter, *multipath*, and *noise*. Clutter refers to unwanted echoes from objects other than the target, such as the ground, trees, or buildings, which can mask or distort the true target signal. Multipath propagation occurs when the transmitted signal reflects off multiple surfaces before reaching the receiver, creating multiple echoes that can cause interference and errors in range estimation. In addition, thermal noise from the receiver electronics and background radio emissions further reduce the quality of the received signal, making reliable detection more challenging [19].

The relationship between the transmitted and received signal powers in a radar system is described by the classical radar range equation. In its simplified form, the radar range equation can be written as

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4 L}, \tag{2.2}$$

where P_r is the received power, P_t is the transmitted power, G_t and G_r are the gains of the transmitting and receiving antennas, respectively, λ is the wavelength of the transmitted signal, σ is the radar cross-section (RCS) of the target, R is the range to the target, and L represents system losses [19]. This equation shows the key factors that affect the strength of the received signal, especially the strong inverse relationship with the fourth power of distance.

Another critical parameter in radar and communication systems is the *signal-to-noise ratio (SNR)*. In radar detection, SNR determines the system's ability to distinguish target echoes from background noise and clutter. Higher SNR values improve detection probability and reduce false alarm rates. In a typical radar system, SNR at the receiver can be defined as

$$\text{SNR} = \frac{P_r}{P_n}, \tag{2.3}$$

where P_n is the noise power, usually modeled as additive white Gaussian noise (AWGN). Achieving and maintaining a sufficient SNR is essential for reliable target detection, especially in environments with significant clutter and multipath effects [19].

A simplified block diagram of a radar system is shown in Figure 2.1, illustrating the main components involved in transmission, target interaction, and reception.

Overall, understanding these fundamental radar principles is essential for evaluating the

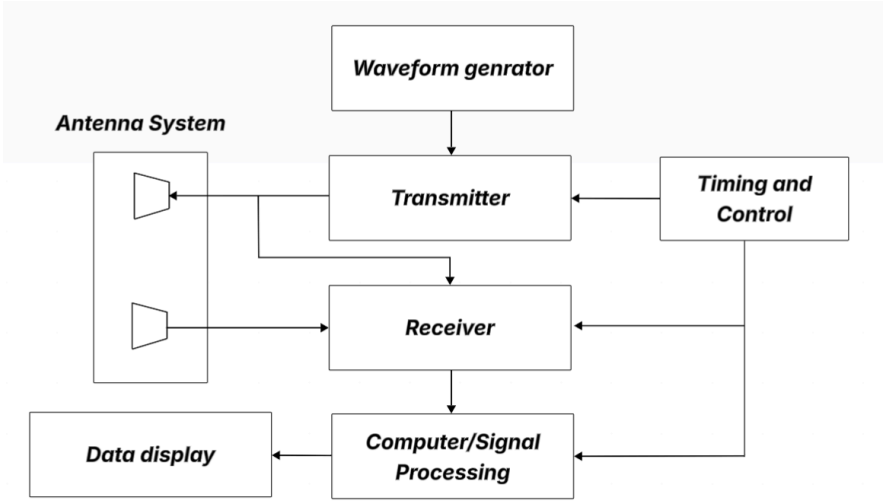


Figure 2.1: Basic radar system architecture showing transmitter, target, and receiver components, from [20].

performance trade-offs and challenges in joint sensing and communication systems, especially within integrated and neuromorphic architectures.

2.1.2 Wireless Communication Basics

Modern wireless communication systems must remain robust against a range of environmental and operational challenges. One of the most significant is **multipath fading**—previously introduced in the context of radar systems— where multiple delayed copies of the transmitted signal can either reinforce or cancel each other when they reach the receiver [21].

The effect of multipath can be mathematically modeled using a linear time-invariant system characterized by an impulse response. A common discrete multipath channel model is given by

$$h(t) = \sum_{l=1}^{L} \alpha_l \delta(t - \tau_l), \tag{2.4}$$

where α_l represents the complex amplitude associated with the l-th path, τ_l is the corresponding delay, and $\delta(\cdot)$ denotes the Dirac delta function. The summation over multiple paths reflects the fact that wireless signals rarely travel along a single, unobstructed line-of-sight path, especially in urban or indoor environments.

To improve resilience to multipath and provide high data rates over short distances, IR and Ultra-Wideband (UWB) technologies have been developed. Unlike conventional nar-

rowband systems that modulate information onto sinusoidal carriers, IR systems transmit extremely short-duration pulses directly into the channel. These pulses occupy a very wide bandwidth, which enables fine time resolution for precise localization and improved robustness to multipath effects [22].

A fundamental modulation scheme used in IR systems is *Pulse Position Modulation* (PPM). In PPM, information is encoded in the time shift of a transmitted pulse relative to a reference time slot. A simple mathematical model for a PPM-modulated IR transmission is given by

$$s(t) = \sum_{l=1}^{L} \phi\left(t - (l-1)T - x_l L_b T_c\right), \tag{2.5}$$

where $\phi(\cdot)$ denotes the transmitted pulse waveform, T is the slot duration, L_b is the bandwidth expansion factor, T_c is the chip time, and $x_l \in 0, 1$ represents the information bit transmitted in the l-th slot [5].

In this scheme, the presence or absence of a time shift within a slot represents a binary '0' or '1'.

Figure 2.2 illustrates the basic principle of PPM signaling, showing how a binary '0' and '1' are mapped to different pulse positions in separate time slots. This approach offers high timing resolution while maintaining low energy per pulse, making it particularly suitable for low-power applications such as wireless sensor networks and ISAC systems.

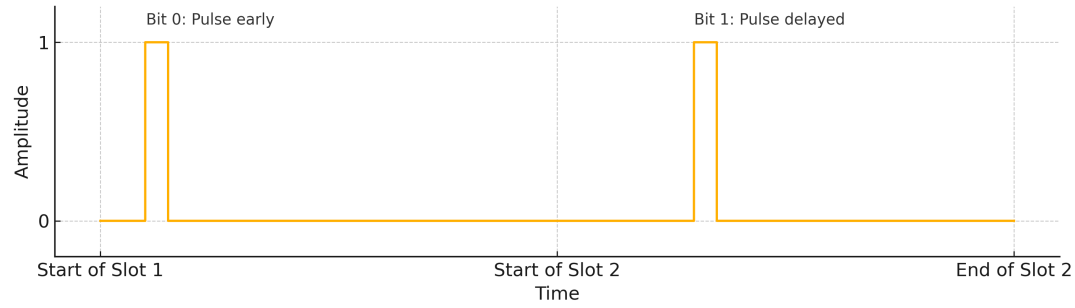


Figure 2.2: Illustration of Pulse Position Modulation (PPM). A binary '0' is represented by an early pulse within the first time slot, while a binary '1' is represented by a delayed pulse within the second time slot.

Understanding multipath propagation, impulse radio signaling, and PPM modulation provides the foundation for designing communication systems that are robust, energy-efficient, and suitable for integration with sensing functions, as explored in the context of neuromorphic ISAC in later sections.

2.1.3 Integrated Sensing and Communication (ISAC)

ISAC is an emerging approach that combines sensing and communication functions within a single system. By sharing resources such as frequency bands and hardware, ISAC allows both tasks to operate simultaneously without unnecessary duplication. This integration improves spectrum efficiency, simplifies system design, and optimizes overall performance [4, 6].

In practice, ISAC systems often face trade-offs between sensing accuracy and communication rates, as maximizing one can limit the other. For example, focusing on high quality sensing may reduce the bandwidth available for communication, while maximizing communication rates could decrease sensing performance. Achieving a balance between these conflicting requirements is essential for the success of ISAC [4**?**]. A practical example of ISAC is a base station that simultaneously performs radar sensing (such as detecting obstacles or estimating distances) and communication tasks (such as transmitting data to users) (Figure 1.1). This dual functionality is expected to be a key feature of future wireless networks, where both high data rates and environmental awareness are critical[4, 5].

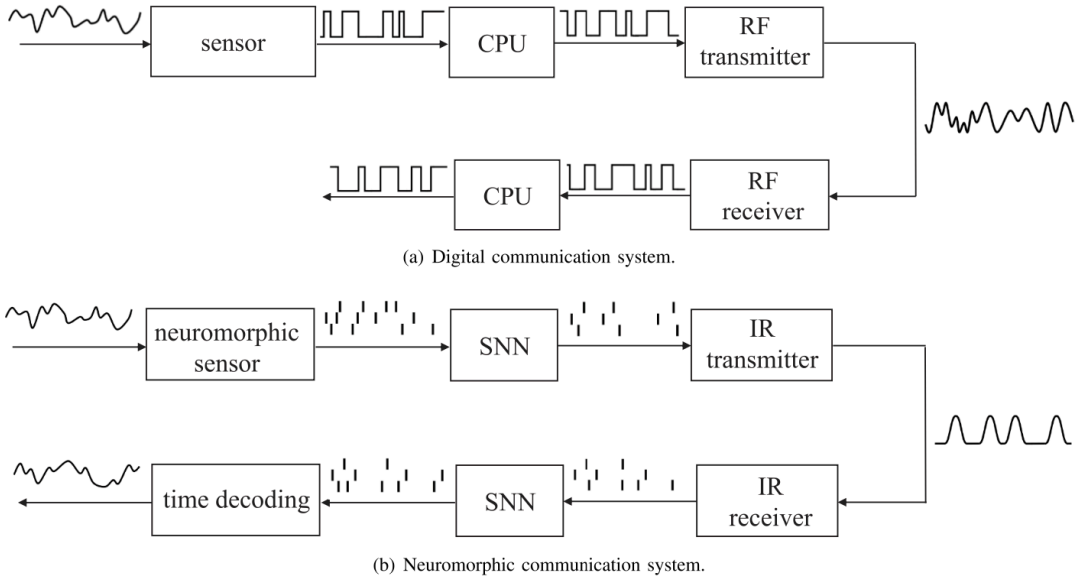
2.1.4 Neuromorphic Computing for Integrated Sensing and Communication (N-ISAC)

Neuromorphic computing, inspired by the functioning of biological brains, offers a new way to improving ISAC systems. Unlike traditional systems that process data continuously, neuromorphic systems use event-driven architectures, where processing occurs only when necessary. This approach greatly reduces power consumption and increases efficiency, making it well suited for applications requiring fast, real time processing.

In ISAC systems, combining IR transmission with neuromorphic receivers offers an effective solution. IR transmits short pulses over a wide frequency range, enabling accurate positioning and strong resistance to signal interference. Neuromorphic receivers process these pulses only when significant events occur, reducing computational load and power consumption. This approach allows systems to efficiently perform both radar sensing and communication tasks without wasting resources [5].

Figure 2.3 contrasts a conventional digital communication system (a) with a neuromorphic communication system (b). While traditional systems rely on continuous data packaging and CPU processing, neuromorphic systems use event-driven spikes and SNNs, which transmit only when meaningful activity occurs, greatly improving efficiency in ISAC scenar-

ios.



(a) Digital communication system.

(b) Neuromorphic communication system.

Figure 2.3: Comparison between (a) a conventional digital communication system and (b) a neuromorphic communication system using SNNs and impulse radio (IR). from [8].

In terms of real-world use, neuromorphic technology is already powering a variety of applications. These range from drone monitoring using Dynamic Vision Sensor (DVS) cameras to brain-computer interfaces and even rapid COVID-19 antibody testing. Key neuromorphic hardware platforms such as Intel's Loihi [16], IBM's TrueNorth [23], and Brainchip's Akida offer a strong base for building energy-efficient systems that are designed for specific tasks and can run well on limited hardware, making them ideal for future ISAC applications [8].

Figure 2.4 illustrates N-ISAC system, where neuromorphic computing processes IR signals for both radar sensing and communication tasks.

2.2 Spiking Neural Networks (SNNs)

Spiking Neural Networks (SNNs) are a special type of neural network inspired by the way biological neurons operate. Unlike conventional neural networks that use continuous signals, SNNs communicate with spikes—discrete events that occur when a neuron's input goes above a certain threshold. This section provides an overview of the key concepts behind SNNs, including their basic principles, different neuron models, training challenges,

Figure 2.4: Illustration of a Neuromorphic Integrated Sensing and Communication (N-ISAC) system from [5].

and the advantages that make them suitable for ISAC applications and real-time processing tasks.

2.2.1 Fundamentals of Spiking Neurons

In biological systems, **neurons** are the basic units of the brain, responsible for processing and transmitting information. Neurons communicate by generating **spikes**, which are discrete electrical pulses that travel along their axons to other neurons. These spikes carry the information the neuron sends to others in the network.

The **membrane potential** is the internal voltage of a neuron, which gradually increases in response to incoming signals from other neurons. When this potential exceeds a certain threshold, the neuron **fires a spike**. After firing, the potential **resets** to its resting state, and the cycle repeats.

The communication process in biological neurons is illustrated in Figure 2.5. As shown in this figure, the neuron consists of multiple parts:

- **Dendrites** receive signals from other neurons.
- The **soma** (cell body) integrates these signals.
- The **axon** transmits the output signal, which is the spike, to other neurons.

The diagram also shows the structure of a biological neuron, where signals from other neurons combined in the soma. Most of the time, neurons remain inactive and only ”fire” when a significant event occurs. This behavior, known as **sparse activation**, makes data storage more efficient. Instead of saving large datasets full of zeros, only the important information

is stored, which saves space. For example, in the binary sequence 0001000000000, only the position of the "1" needs to be stored rather than the entire sequence. This method is called sparse encoding **sparse encoding** [13].

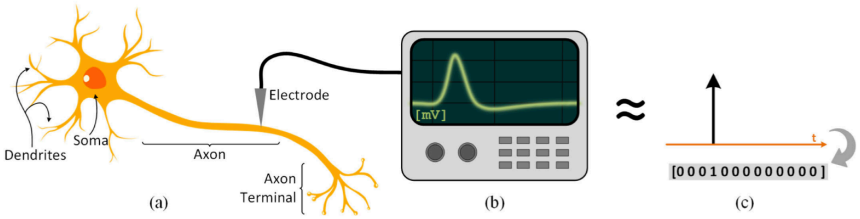


Figure 2.5: (a) Structure of a biological neuron. (b) Representation of a stimulated spike on an oscilloscope, showing the change in potential. (c) Binary spike encoding in SNNs. From [13]

In **event-driven processing**, neurons activate only when a significant input causes their membrane potential to reach a threshold. This is a key feature of SNNs, which process information sparsely and asynchronously. Unlike traditional artificial neural networks (ANNs), where neurons continuously pass data using functions like sigmoid or ReLU, SNNs transmit data as discrete **binary spikes** only when necessary. This leads to lower energy use and makes SNNs especially suitable for real time, low power applications.

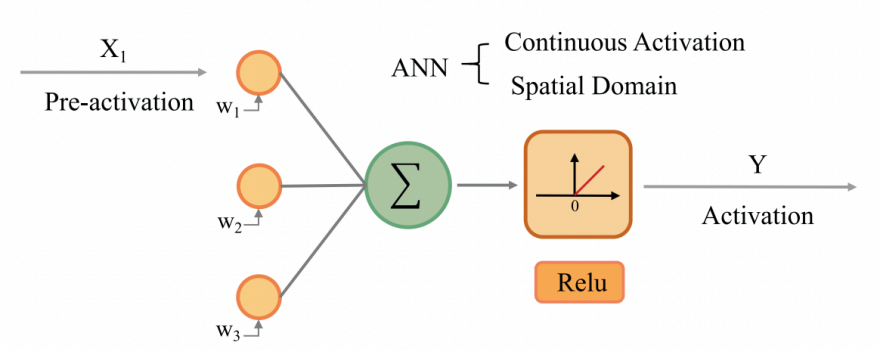


Figure 2.6: Example of an artificial neuron model. From [24]

Figure 2.8 shows the structure of an SNNs neuron. Input spikes are passed to the neuron body through the dendritic tree, and sufficient excitation causes the neuron to emit a spike at the output.

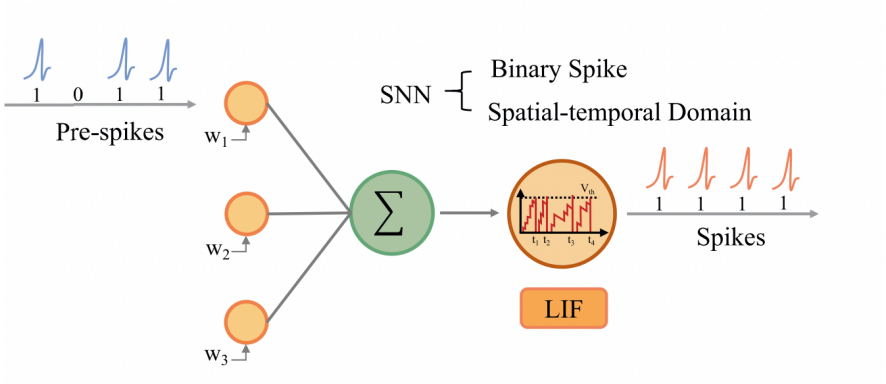


Figure 2.7: Example of a SNNs neuron model. From [24]

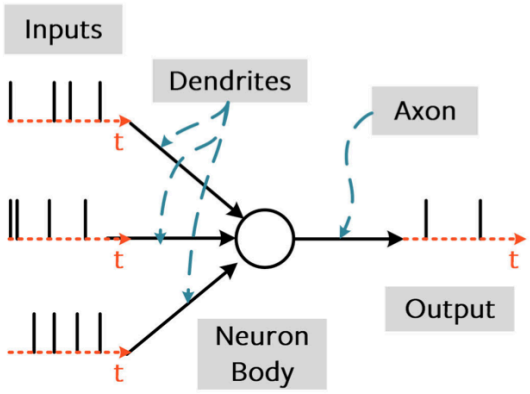


Figure 2.8: Basic structure of an SNNs neuron, showing the flow of spikes from dendrites to axon. From [13]

2.2.2 Spiking Neuron Dynamics and Information Encoding

Spiking neurons are the core components of SNNs and can be modeled in various ways to simulate how real neurons process and transmit information. One widely used model is the Leaky **Integrate-and-Fire (LIF)** neuron. Like an artificial neuron, it sums weighted inputs, but instead of passing them directly through an activation function, it integrates them over time with gradual leakage, similar to an RC circuit. When the membrane potential exceeds a threshold, the neuron emits a voltage spike, as illustrated in Figure 2.9. In this way, information is not carried by the spike itself, but by the timing or frequency of spikes[13].

The membrane potential $U(t)$ in the LIF model is described by a differential equation that captures the dynamic behavior of the neuron. It is given by:

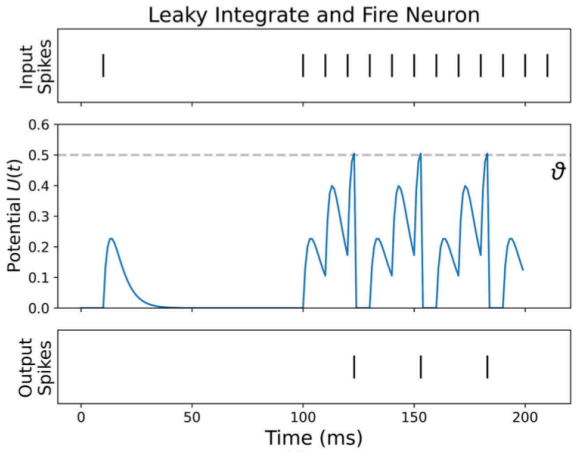


Figure 2.9: Leaky Integrate-and-Fire (LIF) neuron model: Input spikes (top), membrane potential U(t) (middle), which builds up and decays according to the incoming spikes, and output spikes (bottom). From [13].

$$\tau \frac{dU(t)}{dt} = -U(t) + RI_{\text{in}}(t)$$

where τ is the time constant of the neuron, $U(t)$ is the membrane potential, R is the membrane resistance and $I_{\text{in}}(t)$ is the input current at time t.

The membrane potential can also be updated using a discrete method, typically the Euler method, as follows:

$$U[t] = \beta U[t-1] + (1-\beta) I_{\text{in}}[t] \tag{2.6}$$

where $\beta = \exp(-\Delta t/\tau)$, and Δt is the simulation time step [13].

Once the potential exceeds a threshold θ, the neuron emits a spike, which is represented by the following condition:

$$\begin{cases} 1 & \text{if } U[t] > \theta \\ 0 & \text{otherwise} \end{cases}$$

In addition to the LIF model, other spiking neuron models have been developed to capture more complex dynamics. These include:

- **Integrate-and-Fire (IF) Neurons**: A simplified version of the LIF model without leakage, where the membrane potential directly integrates incoming signals.

- **Current-Based (CuBa) Neurons**: These neurons use a current-based approach to model the integration of inputs.
- **Kernel-based Neurons**: These use a kernel function to describe the membrane potential's evolution, adding complexity to the model.
- **Spike Response Model (SRM)**: This model incorporates both synaptic and neuronal feedback using synaptic and feedback filters. The model can be represented by the equation:

$$U(t) = \sum_j w_j \alpha(t - t_j) + \beta(t - t_i) + \gamma_i \tag{2.7}$$

  where:

  - $\alpha(t - t_j)$ is the synaptic filter,
  - $\beta(t - t_i)$ is the feedback filter,
  - γ_i is the bias term.

  The SRM allows for more complex behavior than the LIF model, such as modeling refractory periods and varying synaptic dynamics [5].
- **Generalized Linear Model (GLM)**: An extension of the SRM, the GLM introduces a probabilistic framework for spike generation. Unlike SRM, which uses a hard threshold, GLM applies a sigmoid function to model spike probability, making it suitable for tasks requiring stochastic behavior [5].

Each of these models requires an encoding method to convert sensory data into spike patterns that neurons can effectively process. One widely used technique is rate coding, where the intensity of the input is represented by the frequency of neuron firing. In this method, neurons with a higher input intensity fire more frequently, directly reflecting the strength of the input. As shown in Figure 2.10, the firing rate of a neuron increases with the intensity of the input.

Another method is temporal coding, where the timing of the first spike represents the intensity of the input. In latency coding, neurons fire as soon as they reach a certain threshold, with the first neuron to spike indicating the strongest input. This method is energy efficient because it uses fewer spikes. In population coding, a group of neurons encode the input together. Their combined spike patterns provide a more reliable signal, making this approach more robust and less sensitive to noise. This is illustrated in the population coding diagram, where multiple neurons work together to represent the same data.

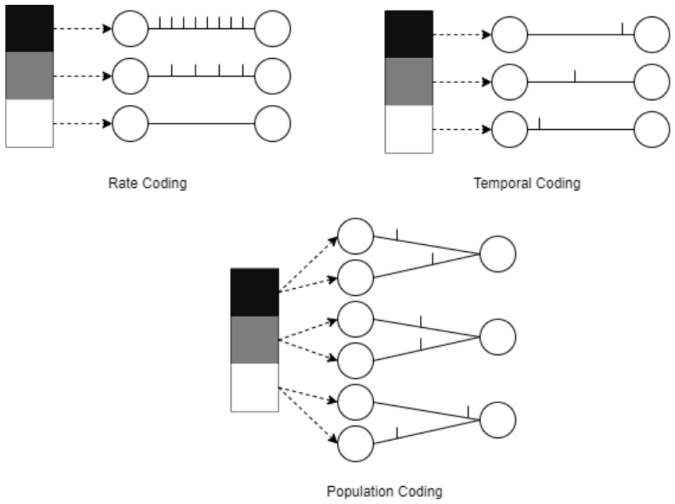


Figure 2.10: Encoding methods in SNNs. From [1]

Decoding strategies are also essential for interpreting the output of an SNNs. Rate-based decoding counts the total number of spikes within a defined time window and assigns the output label to the neuron with the highest spike count. Latency-based decoding relies on the timing of the first spike to decode the intensity of the input These decoding methods align the output with the coding strategy, ensuring efficient real-time information processing.

In this study, we focus on the LIF model, which is particularly suitable for real-time processing in ISAC applications. We apply rate-based decoding, where the neuron with the highest spike count is selected as the output label.

2.2.3 Training SNNs

For SNNs, we use loss functions, activation functions, gradient descent, and backpropagation methods, similar to standard neural networks. However, because spikes are non-differentiable, special techniques like **surrogate gradient** methods and **Backpropagation Through Time (BPTT)** are employed to enable efficient learning in SNNs.

We begin by reviewing some key concepts before addressing the challenges of training SNNs and the solutions to overcome them. In a standard neural network, the **forward pass** processes input data through the network to produce an output, while **backpropagation** adjusts the weights based on the output error. This process involves computing the **gradient** of the loss function with respect to the weights and updating them using an optimization

algorithm, such as **gradient descent**. The weight update is typically expressed as:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w} \tag{2.8}$$

where w represents the weights, η is the learning rate, and $\frac{\partial L}{\partial w}$ denotes the gradient of the loss function L with respect to the weights.

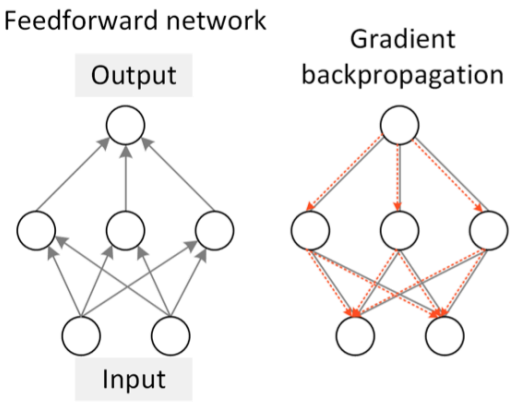


Figure 2.11: Comparison of feedforward network and gradient backpropagation. From [13]

Common loss functions used for training neural networks include:

- **Mean Squared Error (MSE)**, which is widely used for regression tasks and is defined as:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.9}$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the number of data points [25]. MSE measures the average of the squared differences between the predicted and actual values.

- **Cross-Entropy Loss**, which is widely used for classification tasks, especially binary classification, and is defined as:

$$\text{CE}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{2.10}$$

where y is the true label (0 or 1), and $\hat{y}$ is the predicted probability of class 1 [26]. This loss function measures how closely a model's predicted probabilities match the true distribution of labels.

This procedure is standard in both feedforward neural networks SNNs, although SNNs present unique challenges due to the non-differentiable nature of spikes. The non-differentiability of spikes arises from the fact that the derivative of the spike function,

$$\frac{dS}{dU}, \tag{2.11}$$

is zero everywhere except at the threshold θ, where the derivative becomes infinite. In this context, the spike function $S(U)$ is typically modeled as a **Heaviside step function**:

$$H(U - \theta), \tag{2.12}$$

where

$$S(U) = \begin{cases} 0 & \text{if } U < \theta, \\ 1 & \text{if } U \geq \theta. \end{cases} \tag{2.13}$$

As a result, the gradient of the loss function with respect to the weights,

$$\frac{dL}{dW}, \tag{2.14}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial S}\frac{\partial S}{\partial U}\frac{\partial U}{\partial i}\frac{\partial i}{\partial W}, \tag{2.15}$$

can become zero or infinite depending on whether the neuron fires and where the membrane potential lies, as illustrated in Figure 2.12.

One of the main challenges of this behavior is the **dead neuron problem**. This occurs when a neuron fails to fire and, as a result, does not contribute to the overall loss. If the neuron's membrane potential never reaches the threshold, the derivative of the Heaviside function remains zero, and no gradient can flow through that neuron. Consequently, the weights associated with that neuron do not get updated, and it remains inactive, unable to learn or participate in the network's learning process.

One effective solution to this problem is the use of surrogate gradients. **Surrogate gradients** replace the non-differentiable terms in the backpropagation algorithm, enabling continuous weight updates. During the backward pass, a threshold-shifted sigmoid function is commonly used as a surrogate for the Heaviside step function, as shown in Eqs.2.16 and2.17. The sigmoid function provides a continuous approximation of the spike's derivative, allow-

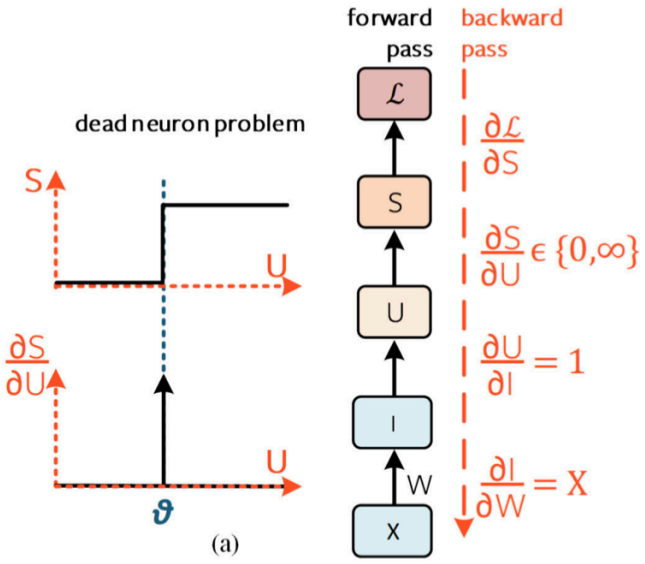


Figure 2.12: Dead neuron problem in SNNs and the gradient flow during the forward and backward passes. From [13].

ing non-zero gradients even when the neuron would otherwise remain inactive. As illustrated in Figure 2.13, this method ensures that all neurons, including those that do not spike, can still contribute to the learning process[13].

$$\sigma(\cdot) = \frac{1}{1 + e^{\theta - U}} \tag{2.16}$$

and therefore

$$\frac{\partial S}{\partial U} \longrightarrow \frac{\partial \tilde{S}}{\partial U} = \sigma'(\cdot) = \frac{e^{\theta - U}}{(e^{\theta - U} + 1)^2} \tag{2.17}$$

The loss function in SNNs can be expressed as shown in 2.18:

$$L = |W_{out} S_{out} - y|. \tag{2.18}$$

The update of the output weightsW_{out}, is given by

$$\frac{\partial L}{\partial W_{out}} = S_{out}. \tag{2.19}$$

To update the input weights W_{in}, the following derivative must be calculated:

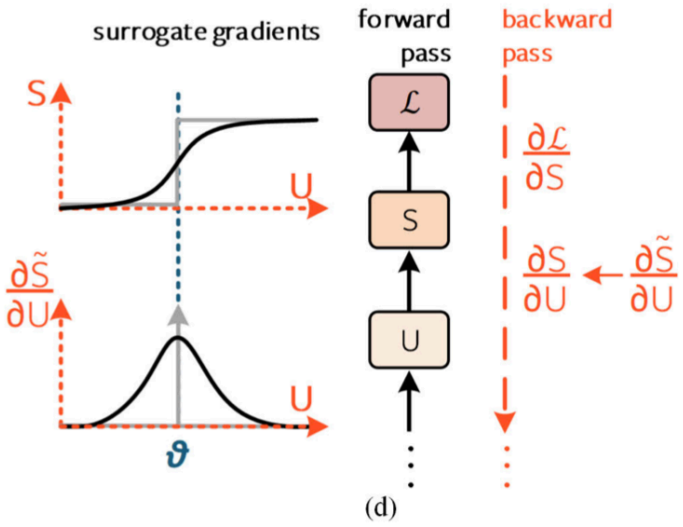


Figure 2.13: surrogate gradients in SNNs. From [13].

$$\frac{\partial L}{\partial W_{in}} = \frac{\partial L}{\partial S_{out}} \frac{\partial S_{out}}{\partial U} \frac{\partial U}{\partial W_{in}}, \tag{2.20}$$

The first term corresponds to W_{out} , as derived directly from the loss function. The second term is generally zero unless surrogate gradients are applied to enable gradient flow through non-spiking neurons. The third term corresponds to S_{in}, as defined in 2.6, where $X = S_{in}$. Together, these terms guide the weight updates during the learning process.

Additionally, because spikes have a temporal nature, another technique commonly used in SNNs is **BPTT**. This method extends the traditional backpropagation algorithm by unfolding the network across time steps, allowing it to capture the sequential behavior of spikes. As a result, BPTT enables the calculation of gradients that consider spike timing, leading to more effective learning in spiking networks [13].

2.2.4 Advantages of SNNs for ISAC

SNNs offer several advantages for ISAC systems. They align well with IR communication„ which is commonly used in low-power wireless communication protocols, such as those based on the IEEE 802.15.4z standard for secure and precise UWB ranging and data transmission. Combining SNNs with IR enables efficient signal encoding and transmission, improving communication performance while maintaining low power consumption [27, 8].

SNNs are also highly suited to real time processing, as they can handle data sequentially

and adapt quickly to changing inputs. This capability is valuable in applications requiring immediate responses, such as remote sensing systems [27, 8]. Additionally, their sparse activations lead to efficient memory usage by storing data only for active neurons, significantly reducing memory demands compared to traditional neural network models.

Finally, integrating SNNs with neuromorphic sensors and communication systems enables seamless end-to-end architectures capable of sensing, transmitting, and analyzing data in real time. This makes SNNs an excellent candidate for energy-efficient and low latency ISAC applications [8].

3

Methodology and Contributions

As neuromorphic computing can process information efficiently when it is encoded as binary signals, known as spikes, we adopt the framework proposed in [5]. To give an overall view of the system, Figure 3.1 shows a neuromorphic ISAC (N-ISAC) system. It has an IR transmitter that modulates digital data using PPM. The signal is then received and processed by a SNNs, which detects the presence of a radar target. This design is ideal for energy-efficient neuromorphic hardware.

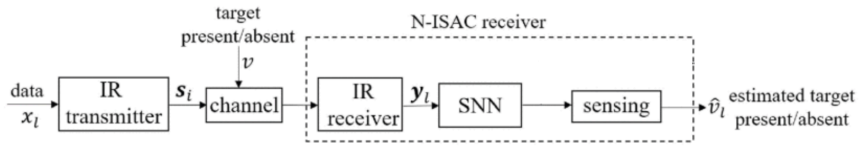


Figure 3.1: Block diagram of an N-ISAC receiver system.

This chapter is divided into three subsections: the ISAC scenario and channel model, which describes how the environment and signals are prepared for testing; the dataset generation process, where the received signals are simulated and formatted for input into the SNNs model; and the design of the SNNs architecture and training process, with two separate branches for classification and regression tasks.

3.1 ISAC Scenario and Channel Model

As shown in Figure 3.1, we consider an N-ISAC system that uses the same IR transmitted signal for both digital communication and radar sensing. In this system, we have L time steps $l = 1, 2, ..., L$ and a bit sequence of information $\{x_1, x_2, ..., x_L\}$, where each bit $x_l \in \{0, 1\}$. The transmitter uses **PPM** with a single antenna to send the data. Each bit is transmitted within a dedicated **time slot** of duration T. So, for L bits, the signal occupies L consecutive time slots.

The transmitted signal is given by:

$$\text{s(t)} = \sum_{l=1}^{L} \phi(t - (l-1)T - x_l L_b T_c) \tag{3.1}$$

where:

- $\phi(t)$ is the pulse waveform,
- T_c is the **chip time** (the smallest time unit),
- T is the **slot time** (duration of each bit),
- L_b is the **bandwidth expansion factor**, determining how many chips fit in a slot.

The number of chips per slot is:

$$T/T_c = 2L_b \tag{3.2}$$

so every time slot contains $2L_b$ chips.

As shown in Figure 3.2, the position of the pulse within the slot encodes the information:

- If $x_l = 1$, the pulse is sent at the **first chip** of the slot, i.e., at position $i * 2L_b$
- If $x_l = 0$, the pulse is sent at the **middle chip** of the slot, i.e., at position $i * 2L_b + L_b$ chips).

This way, the information is represented by the timing of the pulse within the slot, allowing efficient joint sensing and communication.

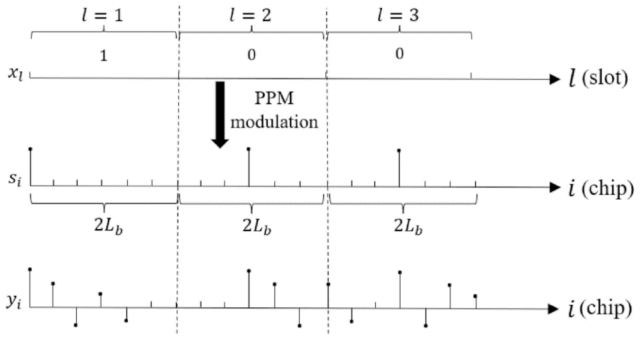


Figure 3.2: Example of PPM showing bit sequence, transmitted signal s_i, and received signal y_i over slots and chips.

The transmitted signal $s(t)$ passes through a **multipath fading channel** to reach the receiver. This channel depends on whether a radar target is present and on reflections from objects **clutter** between the transmitter and receiver.

If a target is present, it reflects the signal after a known **propagation delay** τ_0. The combined effect of the target and clutter on the signal is described by the continuous-time channel response:

$$\mathrm{h(t)} = \nu\beta_0 g(t-\tau_0) + \sum_{c=1}^{N_c} \beta_c g(t-\tau_c) \tag{3.3}$$

where:

- $g(t) = \phi(-t)$ is the **receiver filter**, matched to the transmitted pulse $\phi(t)$,
- β_0 is the **amplitude of the target reflection**, modeled as a complex Gaussian random variable with variance σ_0^2, i.e., $\beta_0 \sim \mathcal{CN}(0, \sigma_0^2)$,
- N_c is the **number of clutter reflections**,
- β_c values represent the **clutter amplitudes**,
- τ_c values are the **delays of the clutter components**.

This model captures the combined effect of a possible target and random clutter to provide a realistic simulation of the channel.

After traveling through the channel, the received signal can be obtained by:

$$y(t) = s(t) * h(t) + z(t) \tag{3.4}$$

where:

- $h(t)$ is the **channel response**,
- $z(t)$ is **AWGN**, to represent a practical wireless environment, with power spectral density N_0,
- $*$ denotes the convolution operation.

The receiver samples $y(t)$ at the **chip rate** $1/T_c$, giving the discrete-time samples:

$$y_i = y(iT_c), \quad i = 1, 2, \ldots, 2L_bL \tag{3.5}$$

For each time slot l, the receiver collects $2L_b$ samples as:

$$y_l = \{y_i\}_{i \in \mathcal{I}_l}, \quad \mathcal{I}_l = \{2(l-1)L_b + 1, \ldots, 2L_bl\} \tag{3.6}$$

The longest delay in the channel should not exceed the symbol time; otherwise, interference between symbols will occur. The channel delay is defined as:

$$T_h = L_hT_c \tag{3.7}$$

where L_h is the number of channel taps. The symbol time (slot duration) is:

$$T = 2L_bT_c \tag{3.8}$$

To avoid interference between symbols, we require:

$$T_h < T \quad \text{or equivalently} \quad L_h < 2L_b$$

The receiver separates the real and imaginary parts of the complex signal y_l into two sequences, forming a vector twice as long as the raw signal:

$$\bar{y}_l = \left[\Re(y_l)^T,\ \Im(y_l)^T\right]^T \tag{3.9}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ represent the real and imaginary components, respectively.

This real vector $\bar{y}_l$ is used as input to the **SNNs** at each slot l, which outputs $\hat{v}_l \in \{0, 1\}$, an estimate of the target presence (1) or absence (0).

3.2 Dataset Generation Process

To train and evaluate the proposed SNNs based N-ISAC system, a custom dataset was generated by simulating the transmitter, channel, and receiver stages. The bandwidth of the system is $B = 400$ MHz, with $L = 80$ slots. Each slot contains $2L_b = 8$ chips, where $L_b = 4$. The chip duration is $T_c = 1/B = 2.5$ ns, and the slot duration is $T = 2L_bT_c = 20$ ns. An upsampling factor of 8 is applied to improve resolution.

Let's begin by explaining the simulation of the transmitted signal, as shown in Figure 3.3.

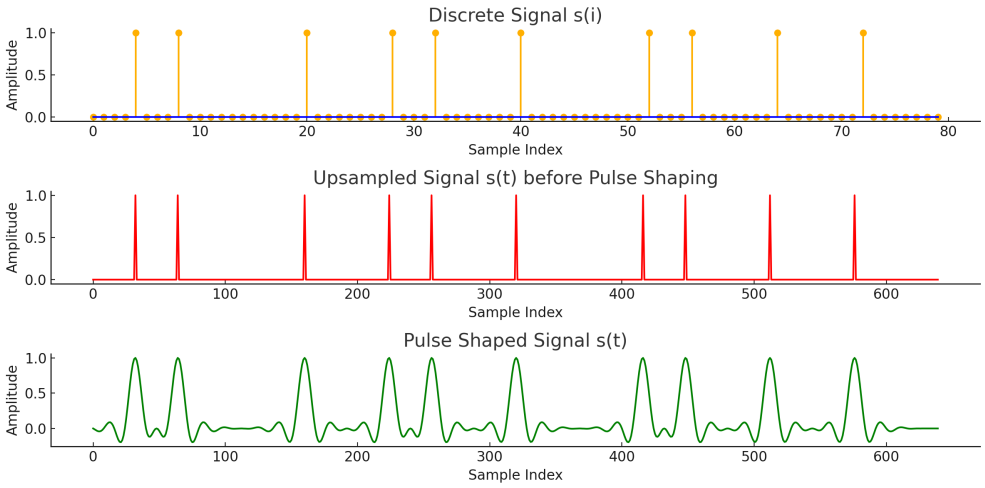


Figure 3.3: Transmitted signal.

First, a random bit sequence $\{x_1, x_2, \ldots, x_L\}$ is created and mapped into a discrete PPM-modulated signal $s(i)$, where each pulse represents a transmitted bit and is placed in its corresponding time slot. The second plot shows the upsampled signal before pulse shaping. In this step, extra zero samples are inserted between pulses to improve time resolution. The third plot shows the final pulse-shaped signal $s(t)$, obtained by filtering the upsampled signal with a raised cosine pulse $\phi(t)$, as defined in Eqn.3.10. This pulse shaping reduces sharp transitions, limits the signal's bandwidth, and helps minimize inter-symbol interference.

$$\phi(t) = \frac{\sin(\pi t/T_s)}{\pi t/T_s} \cdot \frac{\cos(\pi \beta t/T_s)}{1 - (2\beta t/T_s)^2} \tag{3.10}$$

where β is the roll-off factor and T_s is the sampling period. The bit sequence is then modulated onto the pulse-shaped signal using the PPM method.

Figure 3.4 shows the final pulse-shaped signal $s(t)$.

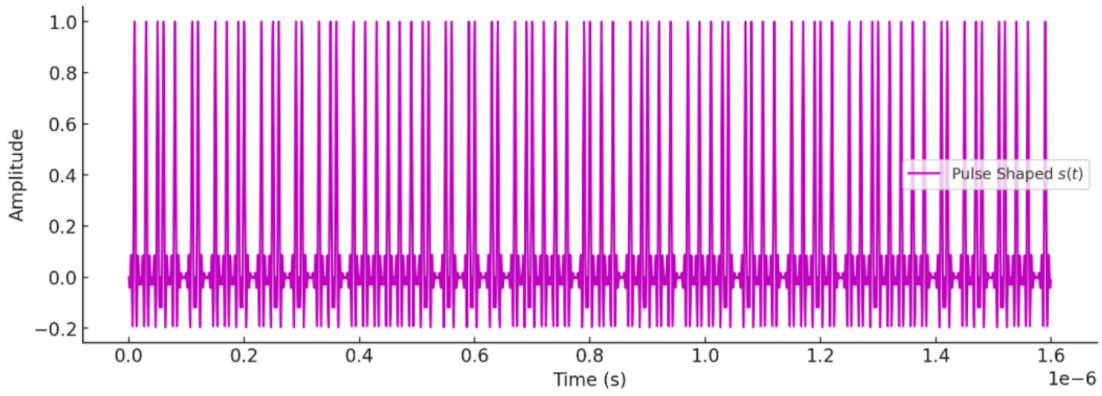


Figure 3.4: final pulse-shaped signal $s(t)$.

The transmitted sparse signal passes through a wireless multipath fading channel modeled as in Eqn. 3.3, where v is a random binary variable indicating whether the target is present ($v = 1$) or absent ($v = 0$).

The target delay τ_0 and clutter delays τ_c are randomly selected within the range $[0, 4T_c]$, with $N_c = 5$ clutter paths. The target reflection amplitude β_0 and clutter amplitudes β_c follow a complex Gaussian distribution with predefined mean and variance.

The channel impulse response $h(t)$ (similar to that shown in Figure 3.5) is applied to the transmitted signal by convolution.

To simulate realistic wireless conditions, AWGN is added to the received signal as expressed in Eqn. 3.4. The noise power is carefully controlled to achieve a specified SNR, which was set to 10 dB in this study.

The energy per bit is assumed to be $E_b = 1$. The effective energy of the channel is computed by first selecting all non-zero values of the channel impulse response $h(t)$, denoted as h_{nz}, and then calculating the mean squared norm:

$$\|h_{\text{nz}}\|^2 = \frac{1}{N} \sum_{i=1}^{N} |h_{\text{nz},i}|^2 \tag{3.11}$$

where N is the number of non-zero channel taps.

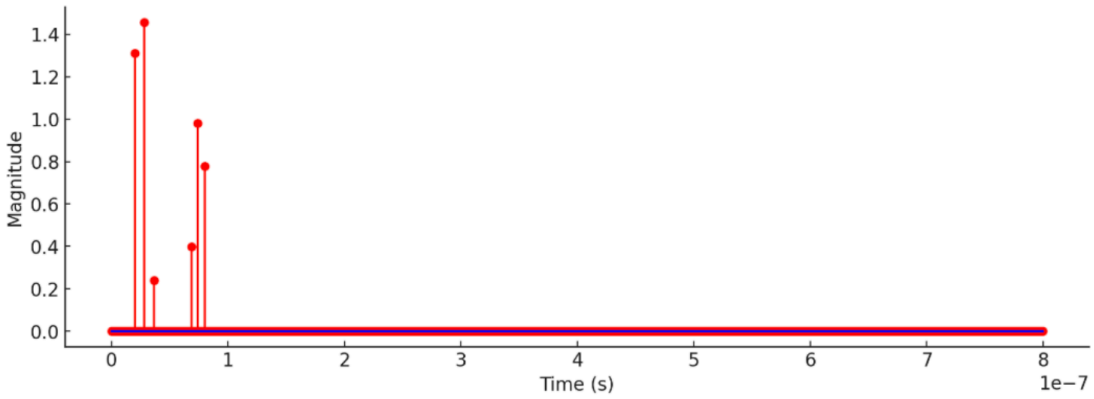


Figure 3.5: Channel Impulse Response $h(t)$, showing spikes for the target return and clutter reflections. Spike positions correspond to delays (τ_0, τ_c) and heights depend on target (β_0) and clutter (β_c) strengths.

The noise spectral density $N_0 B$ is then obtained by:

$$N_0 B = \frac{\|h_{\text{nz}}\|^2 \cdot E_b}{\text{SNR}} \tag{3.12}$$

where SNR is the target signal-to-noise ratio (expressed in linear scale). We model the complex AWGN noise as

$$z(t) = z_{\text{r}}(t) + j\, z_{\text{i}}(t),$$

where the real and imaginary components are independent Gaussian random variables:

$$z_{\text{r}}(t) \sim \mathcal{N}\big(0,\ \sigma\big), \qquad z_{\text{i}}(t) \sim \mathcal{N}\big(0,\ \sigma\big), \quad \text{with } \sigma = \sqrt{N_0 B}.$$

This approach ensures that the noise added to the system is statistically consistent with the desired SNR level.

At the receiver, the signal is downsampled to match the original chip rate. The received signal is segmented into $L = 80$ slots, each containing $2L_b = 8$ samples. For SNNs processing, the real and imaginary parts of each complex sample are split into two separate sequences, as indicated in Eqn. 3.9.

This operation results in a real-valued input vector of size 16 for each slot.

The final dataset is organized as a list of tuples, each structured as:

$$(\bar{y}, \bar{x}, v, \tau_0)$$

where:

- $\bar{y} \in \mathbb{R}^{80\times16}$ is the real-valued matrix of the received signal, with 80 time slots and 16 features per slot (8 real and 8 imaginary values).
- $\bar{x} \in \mathbb{R}^{80\times16}$ is the reference signal (transmitted sequence) in the same format as $\bar{y}$.
- $v \in \{0, 1\}$ is a binary label indicating whether a target is present (1) or absent (0).
- τ_0 represents the propagation delay of the target reflection.

In practical signal processing, estimating the channel or detecting targets requires comparing the received signal to the known transmitted signal. Without the transmitted reference, it would be impossible to invert the channel equation and separate environmental effects (such as multipath fading or reflections) from the transmitted data. Thus, we feed the network both the received signal y and the reference signal x, effectively giving the network the information needed to "reason" about the channel and detect the presence of a target.

As a result, the input to the network is a pair of matrices $(80, 16), (80, 16)$, and the model architecture is designed with two parallel input branches for $\bar{y}$ and $\bar{x}$. Each branch processes its input separately and is then merged into a common feature space, as illustrated in Table 3.1. The architecture is explained in detail in the next section.

Table 3.1: Structure of an input batch showing a sequence of received signals over time.

	$t=0$	$t=1$	$\cdots$	$t=80$
$v=1$	$y_{1,0}$	$y_{1,1}$	$\cdots$	$y_{1,80}$
$v=0$	$y_{2,0}$	$y_{2,1}$	$\cdots$	$y_{2,80}$
$v=0$	$y_{3,0}$	$y_{3,1}$	$\cdots$	$y_{3,80}$
$v=0$	$y_{4,0}$	$y_{4,1}$	$\cdots$	$y_{4,80}$
$v=1$	$y_{5,0}$	$y_{5,1}$	$\cdots$	$y_{5,80}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$v=1$	$y_{128,0}$	$y_{128,1}$	$\cdots$	$y_{128,80}$

3.3 SNNs Architecture and Training

After preparing the dataset, the next step is to design and train the model. The overall architecture of the proposed system is shown in Figure 3.6. The goal of the model is to process the received signal and the reference transmitted signal at each time step to jointly predict two outputs: (1) a classification output for target presence, and (2) a regression output for estimating the target propagation delay. The full architecture and training process are detailed below.

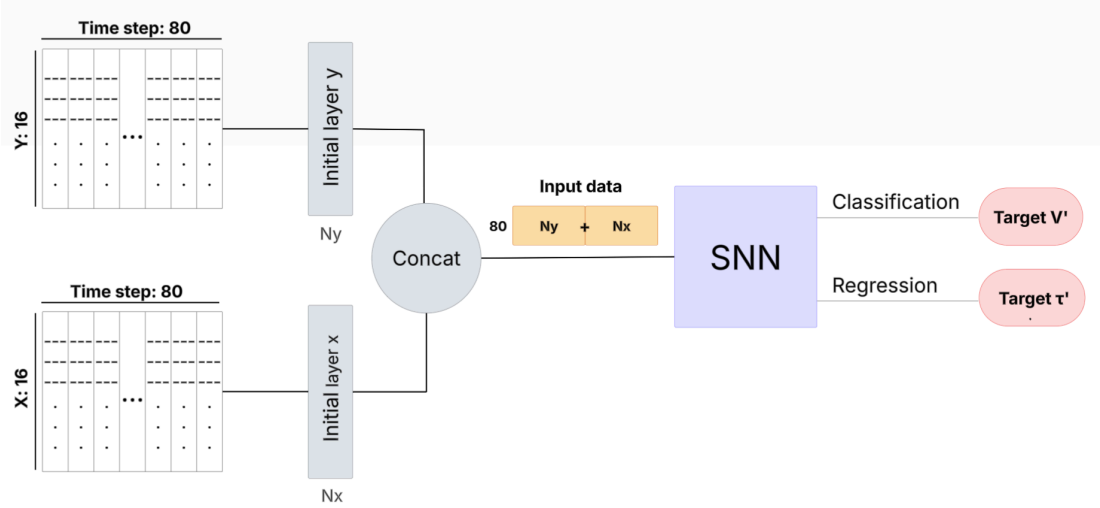


Figure 3.6: The dual-branch SNNs architecture.

3.3.1 Model Architecture

Figure 3.7 illustrates the network's structure. During each training batch (batch size = 128), the model receives two input tensors of shape $(128, 80, 16)$: one for the received signal $\bar{y}$ and one for the transmitted signal $\bar{x}$. Here, 80 represents the number of time slots and 16 the number of features per slot (eight real and eight imaginary components).

Both $\bar{y}$ and $\bar{x}$ are first fed through separate fully connected layers that map their 16-dimensional feature vectors into a 32-dimensional space. This produces two feature streams of shape $(128, 80, 32)$, which are then concatenated along the feature dimension to form a single tensor of shape $(128, 80, 64)$.

The combined sequence is passed into the SNNs core, which consists of 64 LIF neurons. These neurons integrate information over time and emit spikes according to their membrane-

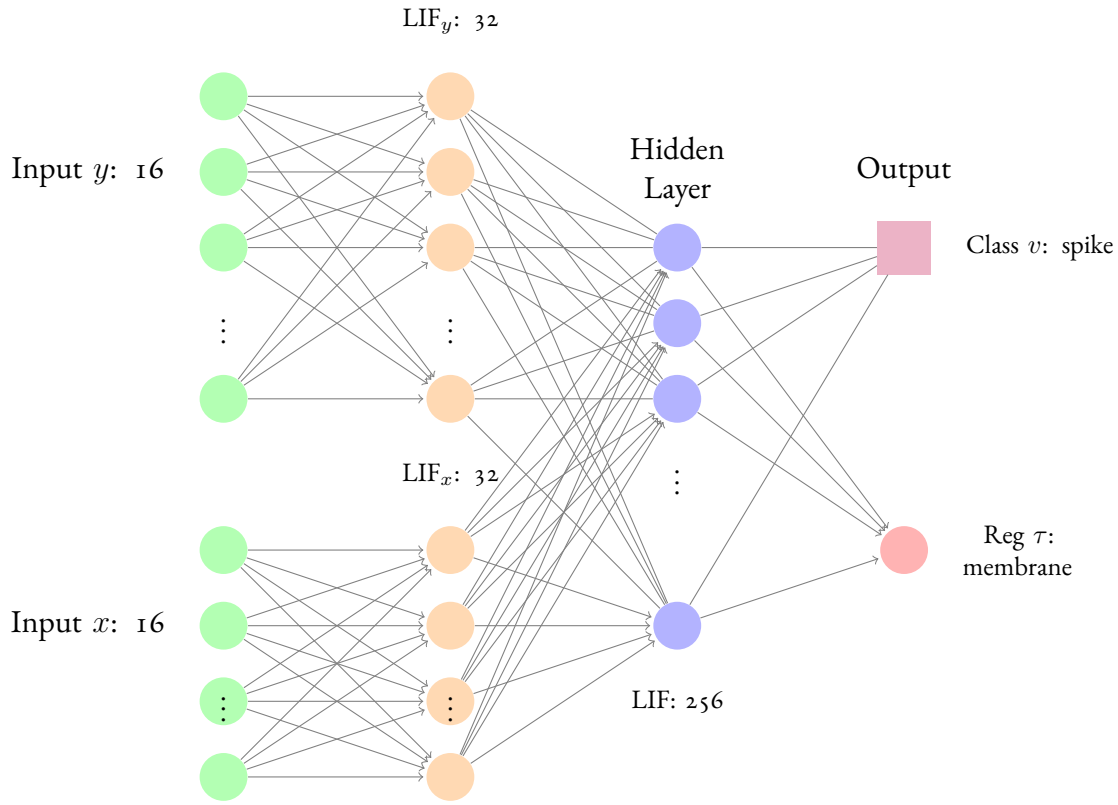


Figure 3.7: Spiking neural network architecture: dual LIF input streams, hidden SNNs layer, and dual outputs for classification and regression.

potential dynamics, capturing the temporal structure of the input signals.

Finally, the spiking outputs are split into two readout branches. The classification branch applies a linear layer followed by a sigmoid activation to produce the probability $\hat{v}$ of target presence. The regression branch uses another linear layer followed by a LIF neuron with reset disabled, meaning its membrane potential does not drop after emitting a spike. By reading this continuously rising potential, the model produces a smooth estimate $\hat{\tau}$ of the propagation delay. This dual-input, dual-output design lets the network compare received and reference signals while effectively separating channel effects from true target echoes.

3.3.2 Training and Loss Functions

The network is trained end-to-end by minimizing a combined loss that addresses both the binary detection of targets and the continuous estimation of propagation delay. For the clas-

sification task, we use the binary cross-entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\big[v \log(\hat{v}) + (1 - v) \log(1 - \hat{v})\big], \tag{3.13}$$

where $v \in \{0, 1\}$ is the ground-truth label and $\hat{v} \in [0, 1]$ is the model's predicted probability of target presence.

For the regression task, we apply the mean squared error loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (\tau_i - \hat{\tau}_i)^2, \tag{3.14}$$

where τ_i is the true delay for sample i and $\hat{\tau}_i$ is the corresponding prediction.

These two objectives are then combined into a single training criterion:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{MSE}}. \tag{3.15}$$

Training is carried out using the Adam optimizer with a learning rate of 0.001. We train for 15 epochs, using a batch size of 128 to balance gradient stability and computational efficiency. The dataset contains $100{,}000$ generated samples, of which 80% are used for training and 20% for testing. Data loading and batching are handled by PyTorch's `DataLoader`. The spiking neural network is implemented in Python 3 using the `snntorch` library.

4 Results

This chapter presents a performance evaluation of the proposed SNNs based N-ISAC system. First, it describes the evaluation metrics used to assess classification and regression performance. Then, it reports on how well the model detects the presence of a radar target and estimates its propagation delay under various conditions. Lastly, it investigates the influence of key channel parameters such as the properties of target and clutter signal on the model's accuracy and robustness.

4.1 Evaluation Metrics

The performance of the model is evaluated using standard classification metrics, including confusion matrix, accuracy, precision, recall, F1 score, ROC-AUC score, as well as regression metrics such as Mean Squared Error (MSE) and Cumulative Distribution Function (CDF) of the absolute error.

The confusion matrix helps visualize the performance of a classification model. It compares the model's predictions with the actual labels. In this matrix, each row shows the real class and each column indicates the class predicted by the model. This format represents four different result types:

- True Positives (TP): The model correctly predicted the positive class.
- True Negatives (TN): The model correctly predicted the negative class.

- False Positives (FP): The model incorrectly predicted the positive class.
- False Negatives (FN): The model incorrectly predicted the negative class.

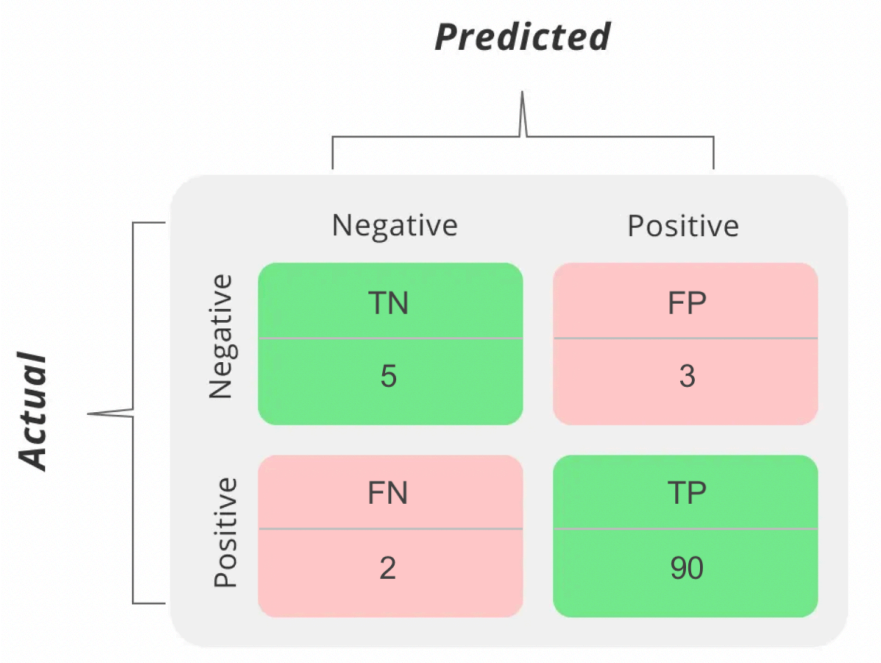


Figure 4.1: Confusion matrix with a total of 100 cases with negative and positive classes [28].

The example in Figure 4.1 covers 100 cases in total, 8 actual negatives and 92 actual positives. The model correctly classifies 5 negatives and 90 positives, but also produces 3 false negatives and 2 false positives. From these counts, we can calculate evaluation metrics such as accuracy, precision, recall, etc.

Accuracy measures the total number of correct classifications divided by the total number of cases and is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

In the example from Figure 4.1, a accuracy of 95% can be calculated.

Precision shows the proportion of predicted positives that are truly positive with the following formula.

$$\text{Precision} = \frac{True\ Positive}{Predictive\ Results} = \frac{TP}{TP + FP} \tag{4.2}$$

In the example from Figure 4.1, a precision of 0.97 (97%) can be calculated.

$$\text{Precision} = \frac{90}{90 + 3} = 0.97 \tag{4.3}$$

High precision means that most items the model labels as positive are truly positive, while low precision shows that many positive predictions are incorrect.

Recall indicates the model's ability to correctly identify actual positive cases. It is defined as

$$\text{Recall} = \frac{True\ Positive}{Actual\ Results} = \frac{TP}{TP + FN} \tag{4.4}$$

From the example in Figure 4.1, a recall of 0.98 (98%) can be calculated.

$$\text{Recall} = \frac{90}{90 + 2} = 0.98 \tag{4.5}$$

High recall means the model finds most of the actual positive cases, while low recall means it misses many of them.

The F1 score, which is the harmonic mean of precision and recall, provides a balance between these two metrics:

$$\text{F1 Score} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{4.6}$$

A high F1 score means the model performs well in both precision and recall, while a low F1 score indicates that one or both of these metrics are weak.

To further evaluate classification performance, the ROC (Receiver Operating Characteristic) curve is used. This plots the true positive rate against the false positive rate at different thresholds. The area under this curve (AUC) provides a single scalar value that reflects the model's ability to discriminate between classes:

$$\text{False Positive Rate} = \frac{FP}{FP + TN} \tag{4.7}$$

For the regression task of estimating the target's propagation delay, the MSE is used to

measeure the average squared difference between predicted and actual delay values:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(p_i - \hat{p}_i)^2 \tag{4.8}$$

where p_i is the true delay and $\hat{p}_i$ is the predicted delay for sample i.

4.2 Performance Results

We used 12 custom designed datasets, each with $100,000$ samples in the tuple list format as $(\bar{y}, \bar{x}, v, \tau_0)$. These datasets were designed to reflect real wireless environments, including effects like multipath fading, clutter, and noise.

The datasets were generated changing important parameters, especially, the mean and variance of the complex channel coefficients for both the target (β_0) and clutter (β_c) components. The configuration of each dataset is shown in Table 4.1, while the remaining system parameters used for dataset generation are summarized in Table 4.2 and discussed in detail in Chapter 3, Section 3.2 Dataset Generation Process.

Then, the model is fed with each dataset to evaluate how well the SNNs can distinguish targets from clutter based on small statistical differences in their signal characteristics. We first develop a baseline model focused only on target detection as a classification task. Then we extend this model by adding a regression branch to estimate the target's propagation delay (τ_0). This extension not only enables the model to predict the physical timing of the target reflection but also helps assess whether classification errors are affected by label distribution, for example if the model tends to favor one class over the other. By combining classification and regression, the model receives richer feedback during training, which improves the overall loss calculation and leads to more accurate predictions. Table 4.3 summarizes the parameters of the SNNs model, which are explained in detail in Chapter 3, Section 3.3 SNNs Architecture and Training.

As shown in Table 4.1, we conducted experiments on 12 custom designed datasets and divided them into two main categories, each with six datasets, based on how the target and clutter channel coefficients differ in their statistical distribution and shift.

In the first category, the clutter paths are closer to the sensing system than the target path. In this scenario, the target distribution is fixed as $\beta_0 \sim \mathcal{CN}(0, 2)$, while the clutter distribution is systematically varied to study how different clutter conditions affect the model's

Table 4.1: Dataset configurations with varying target (β_t) and clutter (β_c) statistics to evaluate model performance.

	Dataset	βt ~ CN(mean ,var)	βc ~ CN(mean ,var)
Clutter Changes	1	mean=0, var=2	mean=0, var=1
	2	mean=0, var=2	mean=0, var=5
	3	mean=0, var=2	mean=0, var=10
	4	mean=0, var=2	mean=1, var=1
	5	mean=0, var=2	mean=2, var=1
	6	mean=0, var=2	mean=3, var=1
Target Changes	7	mean=0, var=1	mean=0, var=2
	8	mean=0, var=5	mean=0, var=2
	9	mean=0, var=10	mean=0, var=2
	10	mean=1, var=1	mean=0, var=2
	11	mean=2, var=1	mean=0, var=2
	12	mean=3, var=1	mean=0, var=2

performance. As summarized in Table 4.4, this setup is divided into two subgroups, Group A and Group B:

- Group A: The clutter mean is fixed at 0, and its variance is increased from 1 to 10, $\beta_c \sim \mathcal{CN}(0, \sigma^2)$ with $\sigma^2 \in \{1, 5, 10\}$.
- Group B: The clutter variance is fixed at 1, and the mean is increased from 1 to 3, $\beta_c \sim \mathcal{CN}(\mu, 1)$ with $\mu \in \{1, 2, 3\}$.

Both classification accuracy and delay estimation error are reported for each setting. These experiments evaluate how much the model is sensitive to clutter strength and distribution shifts, while keeping the target characteristics fixed.

According to Table 4.4, in Group A, test accuracy decreases as clutter variance increases. This means that when the clutter becomes more variable (i.e., uncertain or noisy), the SNN

Table 4.2: System Configuration Parameters

Parameter	**Value**
Number of slots (L)	80
Bandwidth (B)	400 MHz
Bandwidth factor (L_b)	4
Channel duration	100 ns
Channel resolution ($T_c/(2L_b \cdot \text{upsample})$)	$\approx$ 39 ps
Chip time ($T_c = 1/B$)	2.5 ns
Slot duration ($T = 2L_bT_c$)	20 ns
Upsampling factor	8
Transmited signal (xl)	~ Bern(0.5)
Target presence (v)	~ Bern(0.5)
Number of clutter paths (N_c)	5
Clutter delay range (τ_c)	Uniformly distributed $[0, 4T_c]$
Target delay range (τ_0)	Uniformly distributed $[0, 4\,T_c]$
Target coefficient (β_t)	Complex Gaussian distribution ~ CN(mean,σ2)
Clutters coefficient($\{\beta_c\}_{c=1}^{N_c}$)	Complex Gaussian random variables ~ CN(mean,σ2)

model has a harder time distinguishing it from real targets. The increased variability in clutter makes it more difficult to separate target signals from other reflections. As a result, the model is more likely to make mistakes, especially when the clutter characteristics start to overlap with those of the actual target.

In Group B, as the mean of the clutter increases, the strength of the clutter paths becomes similar to or even greater than that of the target. This reduces the separability between the target and clutter in the feature space and leads to lower test accuracy. This trend is visualized in Figure 4.2, which shows the training and validation curves in different clutter settings.

In the second category, the situation is reversed: the clutter distribution is fixed at $\beta_c \sim \mathcal{CN}(0, 2)$, while the target distribution is varied. This setup simulates scenarios where the target is either closer to the sensor or statistically stronger than the clutter. As in the previous case, two subgroups are considered in this category:

- In the first subgroup, the target mean is fixed at 0, while its variance is set to different values in $\{1, 5, 10\}$.
- In the second subgroup, the target variance is fixed at 2, and the mean is increased from 1 to 3.

Table 4.3: SNNs Model Configuration Parameters

Parameter	**Value**
Training data size	$80,000$
Test data size	$20,000$
Batch size	128
Input dimension ($\bar{y}_l$)	80×16
Input dimension ($\bar{x}_l$)	80×16
Input projection size ($N_y = N_x$)	32
Hidden layer size	256
Output size	1 (per head)
Number of time steps	80
Learning rate	5×10^{-4}
Optimizer	Adam
Adam betas	$(0.9, 0.999)$
Weight decay	1×10^{-4}

The results for this configuration are shown in Table 4.5. In Group A, as the target variance increases from 1 to 10, the test accuracy improves significantly, from approximately 67% to 84%. This indicates that when the target reflections become more variable, the SNNs can more easily distinguish them from the fixed clutter background. The wider range of target amplitudes creates clearer patterns in the feature space, helping the network to separate true targets from clutter and thus reducing misclassifications.

In Group B, increasing the mean of target reflections from 1 to 3 also improves performance, with test accuracy rising from approximately 78% to 97%. As the average strength of the target path increases, the target signal becomes more distinguishable from the fixed clutter background. These results confirm that even small changes in the statistical profile of the target (whether in variance or mean) can help the SNN detect targets more reliably. This trend is visualized in Figure 4.3, which shows the training and validation curves in different target settings.

To make the differences easier to compare, Figure 4.4 shows the classification accuracy for both categories (clutter and target) in all datasets.

4.2.1 Impact of Channel Parameters on Classification Performance

To better understand how variations in channel conditions affect target detection, we analyzed the impact of key parameters of the wireless channel, specifically the statistical proper-

Table 4.4: Clutter variation results.

	Dataset Clutter Changes	Train Acc	Train Loss	Train MSE	Test Acc	Train Loss
Group A: Variance	m_t =0 var_t =2 m_c =0 var_c =1	**75.60%**	**50.02**	**3.91**	**75.43%**	**49.77**
	m_t =0 var_t =2 m_c =0 var_c =5	67.58%	55.22	3.53	67.34%	55.38
	m_t =0 var_t =2 m_c =0 var_c =10	64.31%	56.48	3.28	64.26%	56.32
Group B: Mean	m_t =0 var_t =2 m_c =1 var_c =1	67.74%	56.37	4.03	67.50%	56.54
	m_t =0 var_t =2 m_c =2 var_c =1	61.39%	60.19	3.94	61.65%	59.73
	m_t =0 var_t =2 m_c =3 var_c =1	56.87%	60.96	3.94	56.84%	60.98

ties of the target and clutter, including mean and variance of the complex path gains, as well as the relative delay between the target and clutter signals. The goal is to evaluate how statistical separability affects the network ability to detect the target from background reflections.

The three plots in Figure 4.5 illustrate how changes in target and clutter channel parameters affect the test accuracy of the SNNs based detection system.

Plots a and b show how detection accuracy changes when the average strength (magnitude) of the target and clutter paths increases. Four experimental settings are compared: **Clutter Mean Changes** (light blue), **Clutter Variance Changes** (dark blue), **Target Mean Changes** (light purple), and **Target Variance Changes** (dark purple).

We see that accuracy increases significantly with higher target magnitudes, especially when changes come from the target variance or mean. For example, when the mean of the target signal increases from 1 to 3 (purple line), the accuracy improves from 78% to more than 95%. This shows that when the target signal becomes stronger or more distinct, the model can more easily distinguish it from the background clutter.

On the other hand, clutter changes (blue lines) show no such benefit. Despite slight increases in target magnitude (due to random effects), accuracy remains much lower (less than 75%).

The second plot focuses on how increasing the clutter strength affects the accuracy under

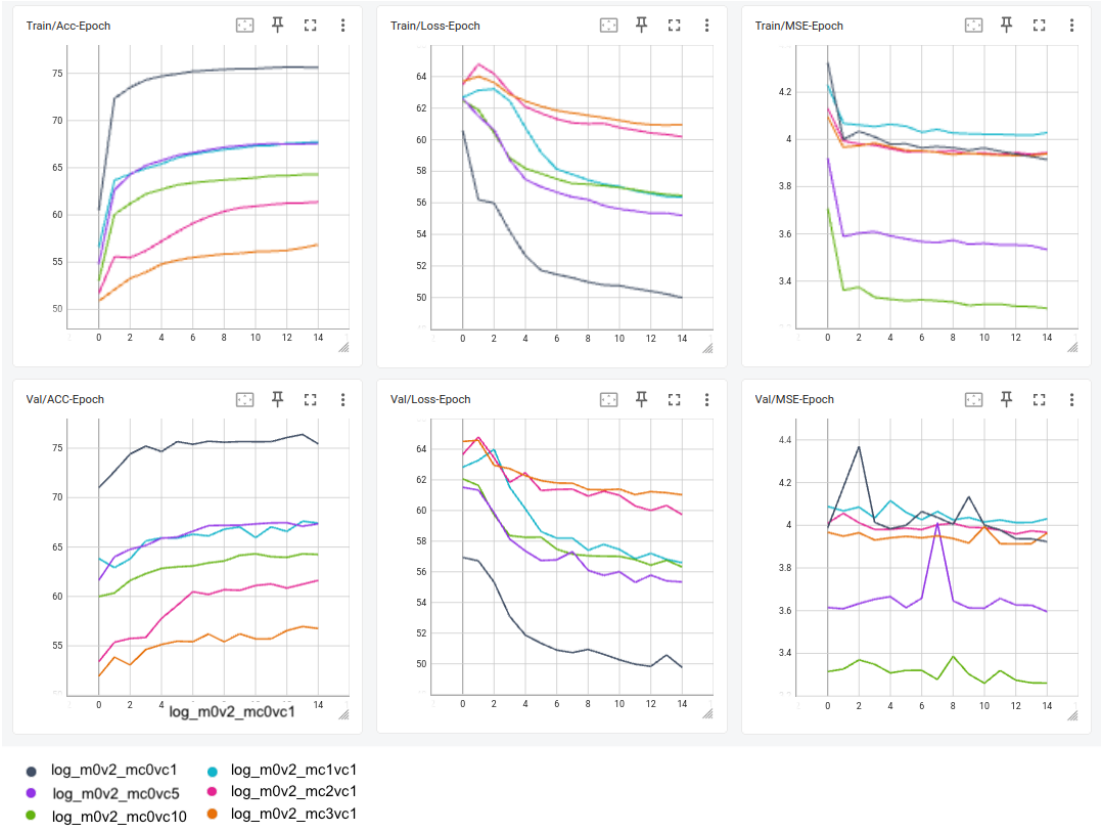


Figure 4.2: Training and validation accuracy, loss, and MSE curves across clutter variation settings. Each line represents a different clutter configuration with the target distribution fixed.

the same four scenarios. Here, the x-axis represents the average magnitude of the clutter paths, and the y-axis is the model's test accuracy.

As expected, a higher clutter magnitude leads to lower accuracy. For both clutter mean and clutter variance changes (orange and red lines), the trend is clear: the more powerful the clutter, the harder it is for the model to detect the target. For example, in the case of clutter variance changes (red), accuracy drops from about 75% to below 65%.

This result emphasizes that the detection model performs best when clutters are weak and the target signal is strong or statistically distinct.

The plot c shows that the changes in the statistical distribution of β_0 (target) and β_c (clutter) affect the amplitude of the reflected signals, not their timing. The delay between target and clutter reflections is determined by their physical positions and the propagation environment, not by the distribution of their channel gains. The delay gap is more affected by random spatial factors than changes in β_0 or β_c.

Table 4.5: Target variation results.

	Dataset Target Changes	Train Acc	Train Loss	Train MSE	Test Acc	Train Loss
Group A: Variance	m_t =0 var_t =1 m_c =0 var_c =2	68.26%	55.88	4.16	67.03%	56.06
	m_t =0 var_t =5 m_c =0 var_c =2	78.13%	46.63	4.16	78.21%	46.22
	m_t =0 var_t =10 m_c =0 var_c =2	84.69%	39.88	4.13	84.72%	39.90
Group B: Mean	m_t =1 var_t =1 m_c =0 var_c =2	78.47%	45.42	4.12	78.63%	45.59
	m_t =2 var_t =1 m_c =0 var_c =2	91.03%	28.86	4.05	90.95%	29.49
	m_t =3 var_t =1 m_c =0 var_c =2	**97.02%**	**18.62**	**3.99**	**97.14%**	**19.19**

For a better understanding of channel impact on classification accuracy, let us look again at its formula, which can be summarized as the sum of one target signal β_0 plus multiple clutter signals $\beta_c^{(i)}$:

$$h = \beta_0 + \sum_{i=1}^{5} \beta_c^{(i)} \tag{4.9}$$

Each β is modeled as a Gaussian random variable:

$$\beta_0 \sim \mathcal{N}(\mu_t, \sigma_t^2), \quad \beta_c^{(i)} \sim \mathcal{N}(\mu_c, \sigma_c^2), \quad \text{i.i.d.} \tag{4.10}$$

Since the clutter signals are independent and additive, the total clutter distribution is also Gaussian with scaled mean and variance:

$$\sum_{i=1}^{5} \beta_c^{(i)} \sim \mathcal{N}\left(5\mu_c, 5\sigma_c^2\right) \tag{4.11}$$

This property is a direct consequence of the additivity of independent Gaussian variables [29].

In the **high accuracy scenario**, the target mean is significantly greater than the clutter

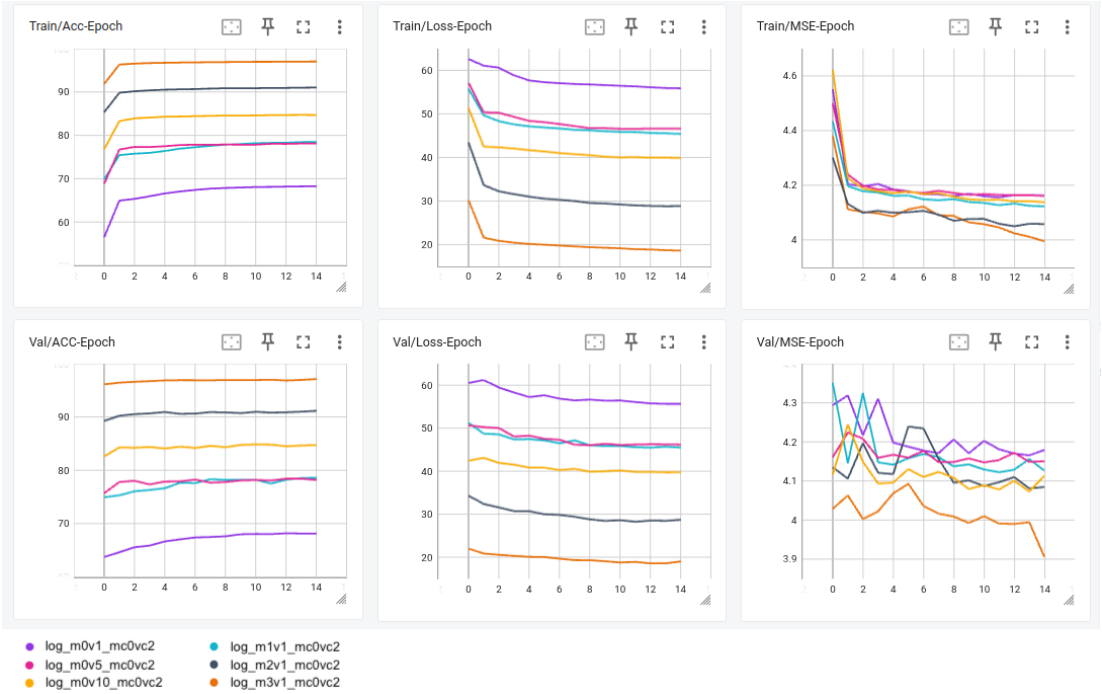


Figure 4.3: Training and validation accuracy, loss, and MSE curves across target variation settings. Each line represents a different target configuration with the clutter distribution fixed.

mean, while the clutter variance is larger. Mathematically:

$$\beta_0 \sim \mathcal{N}(3, 1), \quad \sum_{i=1}^{5} \beta_c^{(i)} \sim \mathcal{N}(0, 10) \tag{4.12}$$

Here, the target stands out as a distinct peak against a noisy, zero-centered clutter background, making classification easier.

In the **low accuracy scenario**, the clutter mean and variance dominate the target:

$$\beta_0 \sim \mathcal{N}(0, 2), \quad \sum_{i=1}^{5} \beta_c^{(i)} \sim \mathcal{N}(15, 5) \tag{4.13}$$

Figure 4.6 represents the distribution of β_0 and β_c for these two cases.

Now, the target signal is buried within a strong clutter wall, making it difficult for the model to distinguish the presence of a target. This statistical dominance by clutter severely reduces the classification performance.

This phenomenon aligns with fundamental principles in **statistical detection theory**,

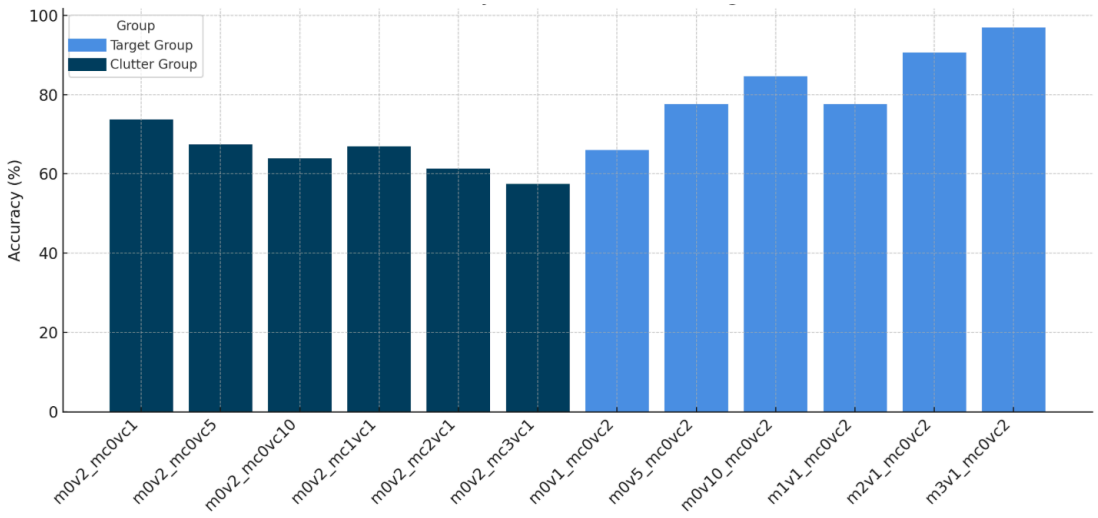


Figure 4.4: Overview of test accuracies for all evaluated datasets, grouped by clutter conditions (dark blue) and target conditions (light blue)

where the ability to detect a target signal depends on the **signal-to-clutter ratio (SCR)**, the relative magnitude and variability between the target and clutter distributions [30, 31].

Our findings illustrate that simply shifting clutter distributions does not necessarily improve classification, especially when the clutter's aggregate signal overwhelms the target. This insight is critical for designing robust detection systems that must operate in environments with multiple interfering clutter sources.
Figure 4.7 supports this observation by providing an overview of the classification accuracy across all 12 datasets, plotted against the distribution of the channel coefficients (β). It highlights how greater statistical separation between target and clutter is strongly correlated with improved model performance.

4.2.2 Detail of the Best Result

Let's take a detailed look at the best-performing dataset ($m_t = 3$, $\text{var}_t = 1$, $m_c = 0$, $\text{var}_c = 2$). As shown in Figure4.8 (a), both the training and testing MSE curves improve continuously and end with values around 4.00 for training and 4.04 for testing. The small and consistent difference between them shows that the model generalizes well and has low variance across epochs.

The combined loss from classification and regression drops quickly in the first few epochs and then becomes more stable. This means that the model learns efficiently in the early stages

and then improves gradually. The training and testing loss follow each other closely, suggesting that there is no overfitting.

In the accuracy curves, training and testing accuracy stay close to each other throughout the process. Interestingly, the test accuracy is slightly higher than the training at the beginning, which might be a sign of helpful regularization. In the end, the test accuracy reaches about 97%, confirming that the model has high classification performance.

As shown in the CDF of the absolute errors (Figure4.8 (d)), about 40% of the predictions have an error smaller than 4.03, around 85% are below 4.13, and almost all predictions are under 4.30. This means that although there is some spread in the errors, the model is generally consistent and does not make large mistakes when estimating delay.

The confusion matrix in Figure 4.9 illustrates the performance of the model. Out of the 9922 actual negatives, only 36 are incorrectly labeled as positives (false positives). Similarly, out of 10046 actual positives, it misses 580 (false negatives). This proves that the model can distinguish targets from non-targets with remarkable accuracy.

A precision of 0.9962 means that nearly every time it predicts a target, it is correct. Meanwhile, a recall of 0.9423 shows that it successfully detects more than 94% of all actual targets. The ROC curve also confirms this performance, staying close to the top-left corner and giving an AUC of about 0.99. In other words, at any decision threshold, the model clearly separates targets from clutter.

4.2.3 Analysis of Accuracy versus Target–Clutter Delay

The delay-based accuracy plot (Figure4.10) is essential for understanding the limitations of the system's temporal resolution. By analyzing the classification accuracy as a function of the target–clutter delay ($\Delta\tau = \tau_0 - \tau_c$), we can directly assess the system's ability to distinguish closely spaced echoes, which is particularly important in practical scenarios with dense clutter or multipath. This analysis reveals the specific time intervals at which the system begins to reliably distinguish the target from the clutter, highlighting the transition from pulse overlap (where discrimination is difficult) to fully resolved echoes (where discrimination should be easier). The target–clutter delay is calculated as the minimum temporal separation between the target and the nearest clutter, scaled by the resolution parameter of the system. The red dashed line indicates the system's time resolution at 0.039 ns. When the target–clutter delay is very small (0.0–0.4) ns, the accuracy is about 90.3 %, indicating significant overlap and pulse interference. As soon as the delay moves into the (0.4–0.8) ns range, the accuracy jumps to 98.8 %, since the pulses are now separated beyond the system's resolution and can

be clearly distinguished. In the next bin (0.8–1.2) ns, accuracy remains high at 98.4 %, as expected.

Surprisingly, in the 1.2–1.6 ns bin, accuracy drops to 92.2 %, even though the echoes are well separated. This anomaly is likely due to the relatively small number of samples in this category (991 out of 20′000; see Figure 4.11), which makes the accuracy measurement more sensitive to statistical fluctuations. Another contributing factor may be the random amplitudes (β_0 and β_c) resulting from Gaussian distributions: rare cases where the clutter amplitude is relatively strong compared to the target could cause misclassification, even at larger delays. Moreover, since the sample distribution is strongly skewed towards the (0.4–0.8) ns region, the classifier may be less robust for delay values that are less frequently represented during training. To further clarify this effect, it is recommended to collect more data specifically in the 1.2–1.6 ns delay range and to analyze the amplitude ratios and outlier events in this bin. Finally, in the last bin (1.6–2.0 ns), the accuracy improves again to 98.6 %, likely reflecting a return to statistical stability and effective pulse separation at large delays.

Table 4.6: Accuracy vs. Target–Clutter Delay

$\Delta\tau$ (ns)	Samples	Accuracy
$0.0 - 0.4$	3,734	0.903
$0.4 - 0.8$	13,074	0.988
$0.8 - 1.2$	1,538	0.984
$1.2 - 1.6$	991	0.922
$1.6 - 2.0$	663	0.986
$2.0 - 2.3$	0	NaN

4.2.4 Analysis of Model Robustness Across SNR Levels

To evaluate the robustness of the trained model, we generated five test datasets under varying noise conditions, with 5 db steps, from 0 to 20 dB SNR.

Table 4.7 shows how the classification accuracy of the above trained model changes when tested on datasets with different SNR levels.

At 0 dB SNR, accuracy falls to around 74 %, which makes sense because the noise is as strong as the signal and the target echo is mostly hidden. When we increase SNR to 5 dB, accuracy jumps by roughly 17 %, showing that even a small reduction in noise greatly helps detection. From 5 dB to 20 dB, accuracy improves a bit more but then remains constant at

Table 4.7: Model accuracy across SNR Levels.

Trained Model	SNR	Accuracy
$dt\_trg\_m3.path$	0 dB	74.13%
	5 dB	91.17%
	10 dB	94.04%
	15 dB	93.87%
	20 dB	93.70%

around 94 %. Above 10 dB, further noise reduction brings little benefit, meaning the model has already extracted all the useful signal features it can.

In other words, the model is robust across moderate to high SNRs (5–20 dB), maintaining about 94 % accuracy. However, at very low SNRs (< 5 dB), its performance drops significantly, which making it less reliable in extreme noise conditions.

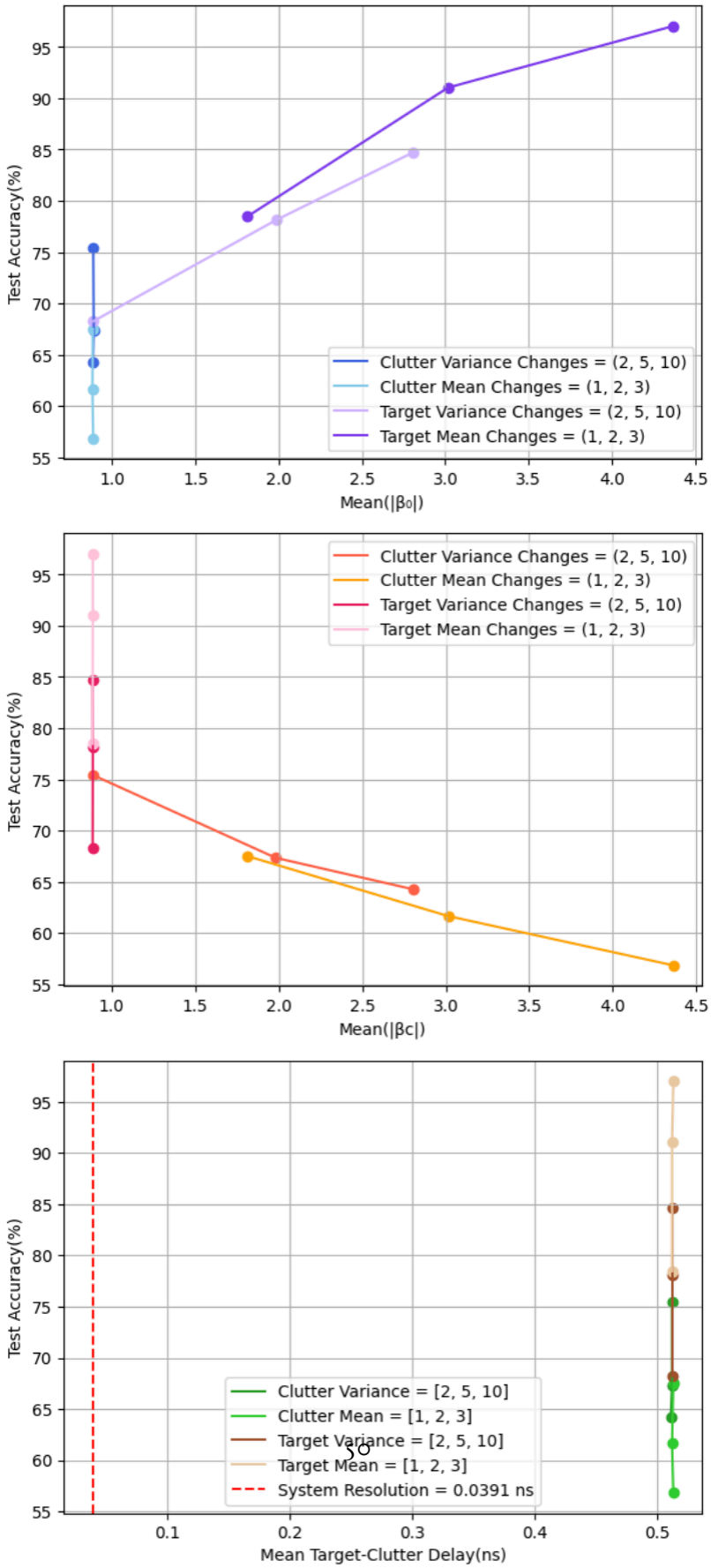


Figure 4.5: a,b)Test accuracy vs magnitude of β_0 and β_c, c)Test accuracy vs target–clutter delay, for all datasets

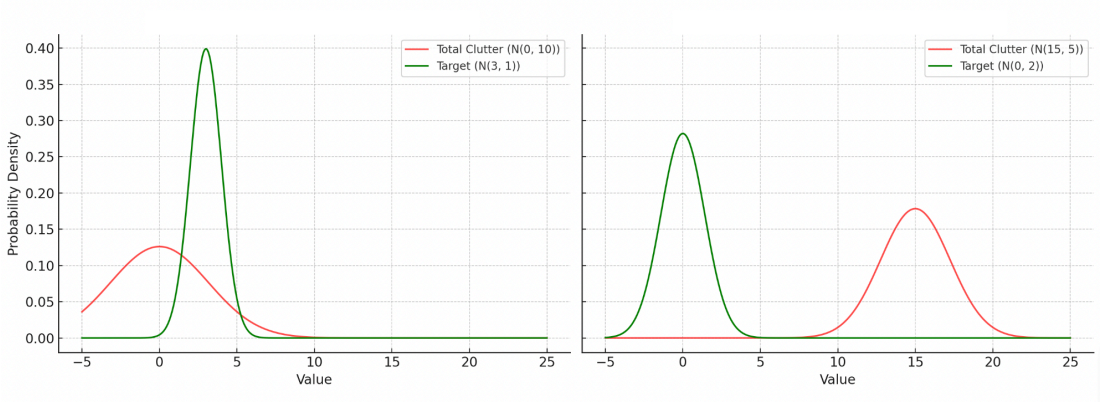


Figure 4.6: Target and clutters coefficients. a) Case A: Target Strong, Clutter Weak, b) Case B: Target Weak, Clutter Strong

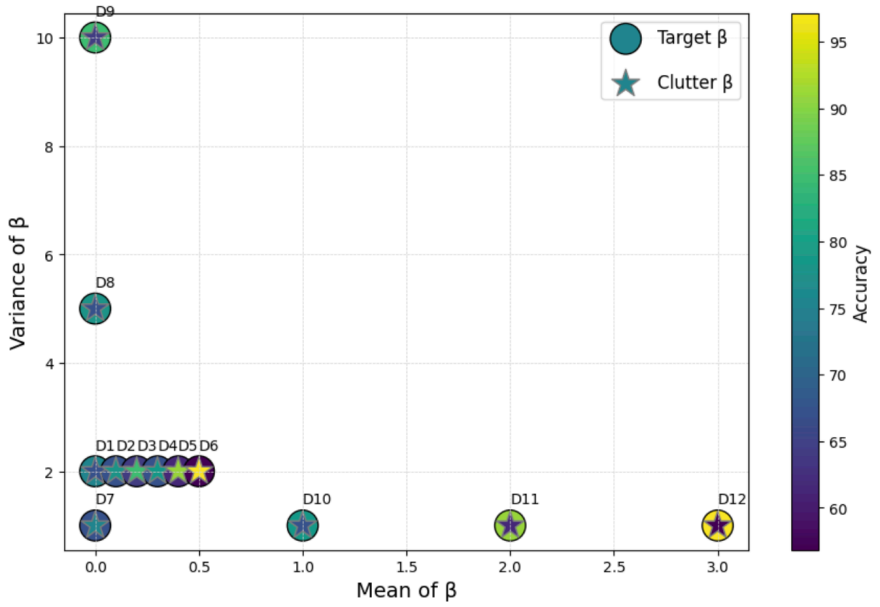


Figure 4.7: Accuracy vs Distribution of β.

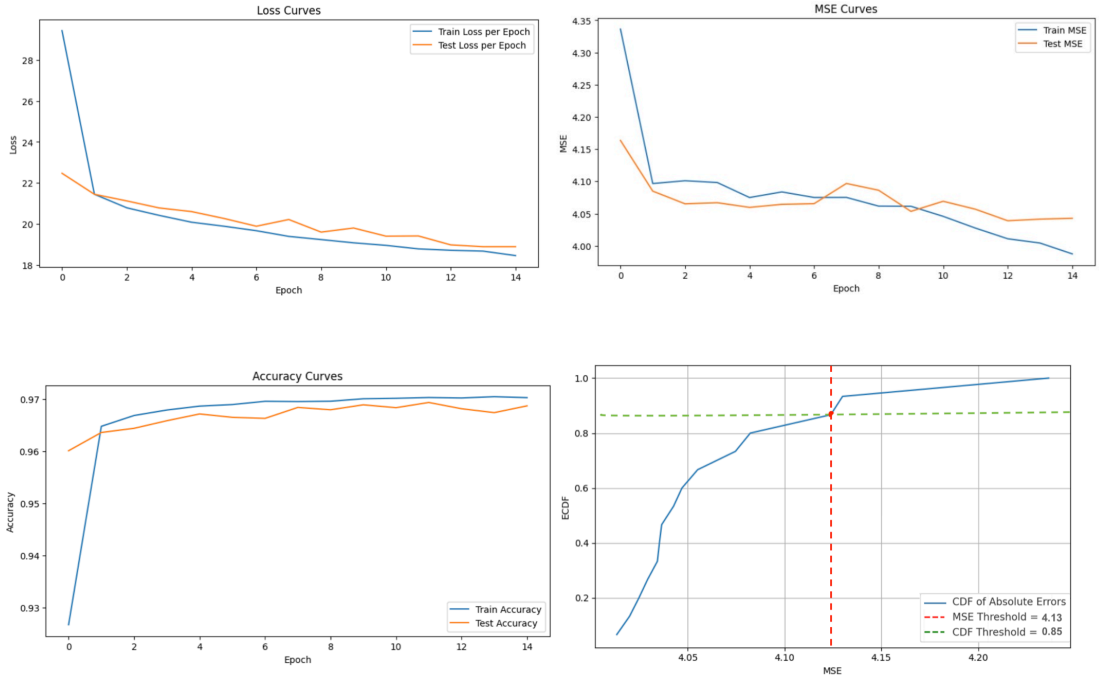


Figure 4.8: Training and test MSE, loss, accuracy and CDF curves over 15 epochs for the best-performing dataset ($\beta_0 \sim \mathcal{CN}(3,1),\ \beta_c \sim \mathcal{CN}(0,2)$).

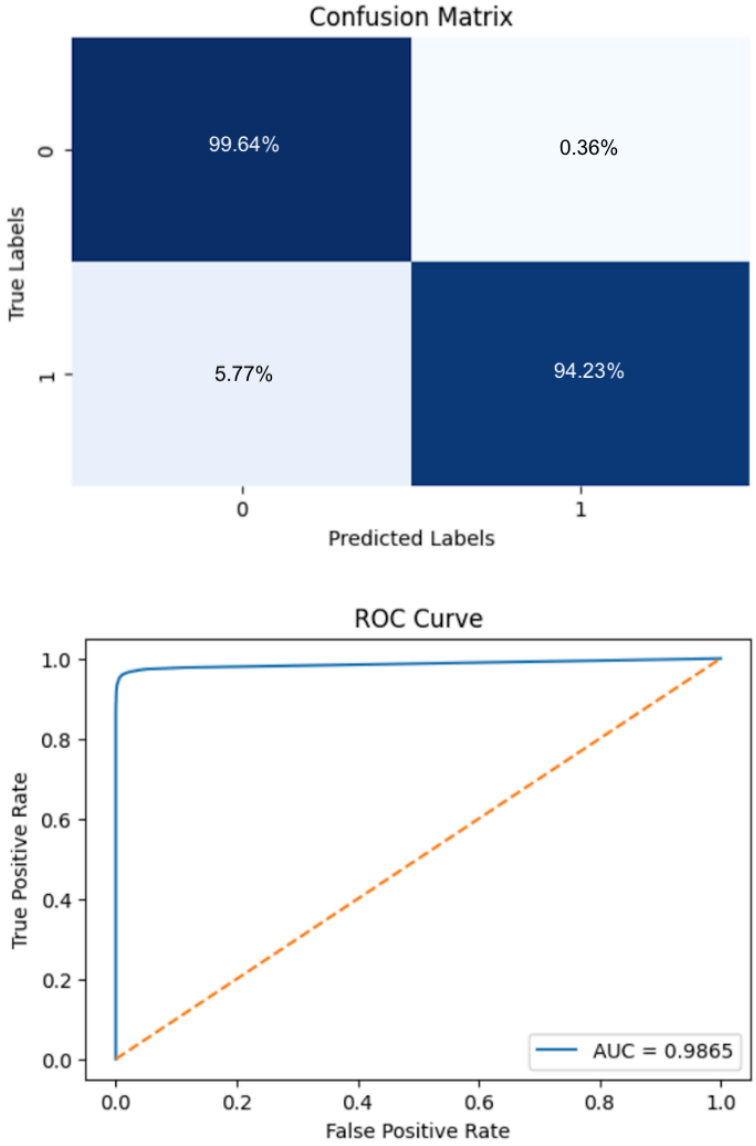


Figure 4.9: Confusion matrix and ROC curve for the best configuration, showing Precision = 0.9962, Recall = 0.9423, F1 Score = 0.9685, and ROC-AUC Score = 0.9865.

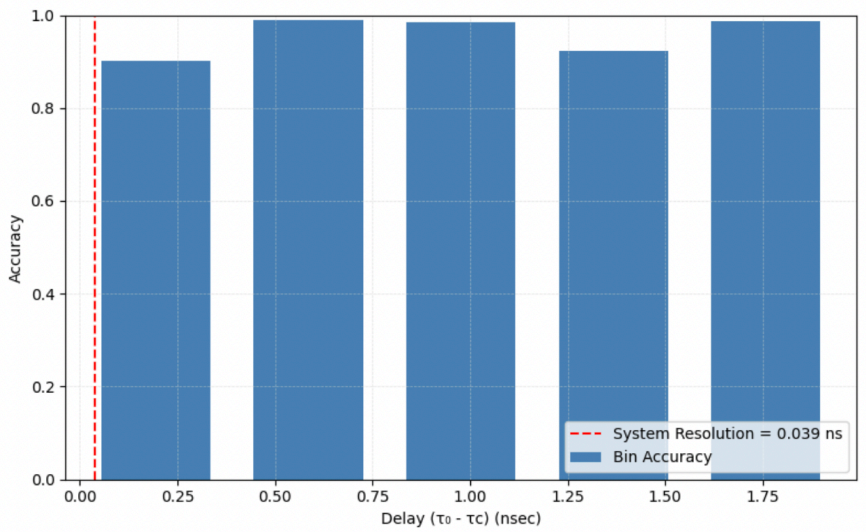


Figure 4.10: Test accuracy vs target–clutter delay for the best configuration.

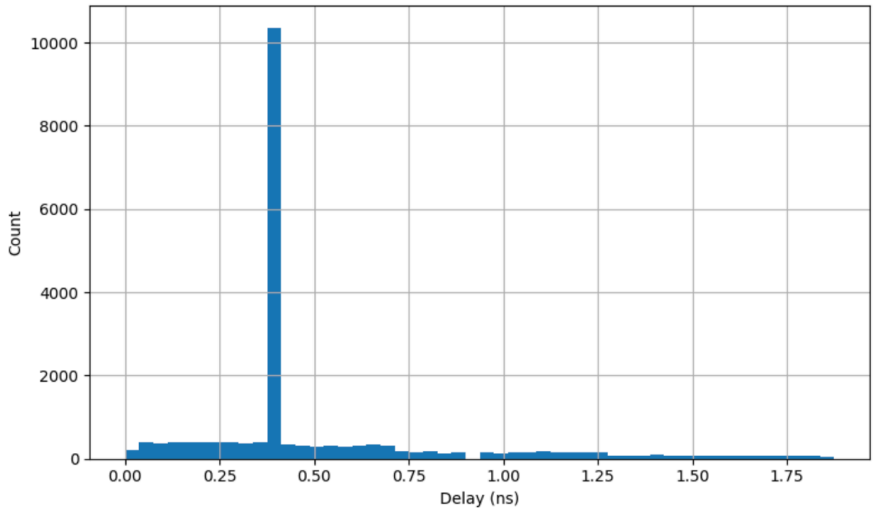


Figure 4.11: Histogram of Target–Clutter Delays ($\Delta\tau$) in the Test Dataset.

5 Conclusion

This thesis presented a new dual-branch SNN architecture that jointly performs target detection (classification) and propagation delay estimation (regression) using pulse position modulated (PPM) impulse radio signals in the context of ISAC systems.

with an extensive review of ISAC systems, specifically focusing on the integration of communication and sensing in the context of radar and wireless communications. It discusses current methodologies and the progress in using SNNs for tasks traditionally handled by DNNs. A detailed investigation into the advantages of neuromorphic computing and SNNs for ISAC tasks was carried out, then presented the methodology and design choices for the proposed dual-branch SNN architecture, which performs simultaneous target detection and propagation delay estimation.

To evaluate the performance of this model, 12 custom datasets were generated to emulate realistic wireless channels and examine how the SNN behaves under different statistical configurations of channel coefficients. Experimental results show that the proposed model achieves high precision and recall, particularly when the statistical characteristics of target reflections and clutter are sufficiently distinct. These experiments explore how statistical separability from the target side influences performance and validate the SNN's ability to learn subtle differences between target and clutter signals. By comparing how accuracy changes across different configurations, we can assess the sensitivity of the model to subtle or strong shifts in the statistical separability between target and clutter components. Additionally, the dual-branch design enables simultaneous classification (target presence) and regression

(propagation delay estimation), showing that spiking models are suitable for multi-objective learning in ISAC scenarios.

Despite these outcomes, several limitations remain. First, since training and evaluation are entirely rely on generated datasets, the model's ability to generalize to real-world environments has not yet been validated. The absence of empirical measurements causes uncertainty about how the SNN would perform in practical deployments. Second, the model's sensitivity to small variations in signal features could reduce its robustness when faced with highly dynamic or unpredictable channel conditions.

To address these challenges, future work should focus on integrating and validating the proposed SNN on real-world hardware using measured wireless channel data. This step is crucial to confirm the performance under practical conditions. In addition, incorporating more complex and dynamic environmental models such as time-varying clutter and expanding the architecture to support multi-target detection can increase realism and improve the robustness of the system for real-world ISAC applications.

References

[1] O. Shears and A. H. Yazdani, "Spiking neural networks for image classification," *Advanced Machine Learning*, pp. 1–7, Dec 2020, image from Fig. 3, depicting rate, temporal, and population coding in SNNs. [Online]. Available: https://www.researchgate.net/publication/349370733_Spiking_Neural_Networks_for_Image_Classification

[2] D. Cohen, K. V. Mishra, and Y. C. Eldar, "Spectrum sharing radar: Coexistence via xampling," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 3, pp. 1279–1296, 2017.

[3] L. Zheng, M. Lops, Y. C. Eldar, and X. Wang, "Radar and communication coexistence: an overview," *arXiv preprint arXiv:1902.08676*, 2019.

[4] F. Liu, Y. Cui, C. Masouros, P. Grant, A. P. Petropulu, and I. Elder, "Integrated sensing and communications: Toward dual-functional wireless networks for 6g and beyond," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 6, pp. 1728–1767, 2022.

[5] J. Chen, N. Skatchkovsky, and O. Simeone, "Neuromorphic integrated sensing and communications," *IEEE Wireless Communications Letters*, vol. 12, no. 3, pp. 476–480, 2023.

[6] J. B. Evans, "Shared spectrum access for radar and communications (ssparc)," *DARPA, Press Release.[Online]. Available: http://www. darpa. mil/program/shared-spectrum-access-for-radar-and-communications*, 2016.

[7] J. A. Zhang, M. L. Rahman, K. Wu, X. Huang, Y. J. Guo, S. Chen, and J. Yuan, "Enabling joint communication and radar sensing in mobile networks—a survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 306–345, 2021.

[8] J. Chen, N. Skatchkovsky, and O. Simeone, "Neuromorphic wireless cognition: Event-driven semantic communications for remote inference," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 2, pp. 252–267, 2023.

[9] NYU Wireless, "Integrated sensing and communication (isac)," https://wireless.engineering.nyu.edu/integrated-sensing-and-communication-isac/, 2023, accessed: 2025-04-26.

[10] C. Wang, A. N. Haque, and T. Taleb, "Vehicle-to-everything (v2x) services supported by lte-based systems and 5g," *IEEE Communications Standards Magazine*, vol. 4, no. 3, pp. 22–28, 2020.

[11] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[12] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6g be?" *Nature Electronics*, vol. 3, no. 1, pp. 20–29, 2020.

[13] J. Eshraghian, M. Ward, E. O. Neftci *et al.*, "Training spiking neural networks," *Nature Machine Intelligence*, vol. 5, no. 1, pp. 18–32, 2023.

[14] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[15] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press, 2002.

[16] M. Davies and L. S. R. E. M. L. S. P. M. R. K. S. P. P., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 1095–1108, 2018.

[17] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–36, 2019.

[18] E. Neftci and J. Eshraghian, "Snntorch documentation," https://snntorch.readthedocs.io/en/latest/readme.html, 2022, accessed: 2025-04-26.

[19] M. I. Skolnik, *Radar Handbook*, 3rd ed. New York, NY, USA: McGraw-Hill Education, 2008.

[20] Best Engineering Projects, "Basic radar system block diagram," https://bestengineeringprojects.com/basic-radar-block-diagram/, accessed: 2025-04-28.

[21] S. Jakborvornphan, "Fading characteristics over wireless channels," *Journal of Engineering and Applied Sciences*, vol. 15, no. 2, pp. 444–451, 2020.

[22] S. Migla, K. Rubuls, N. Tihomorskis, T. Salgals, O. Ozolins, V. Bobrovs, S. Spolitis, and A. Aboltins, "Ultra-wideband analog radio-over-fiber communication system employing pulse-position modulation," *Applied Sciences*, vol. 15, no. 8, p. 4222, 2025.

[23] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[24] X. Liao, Y. Wu, Z. Wang, D. Wang, and H. Zhang, "A convolutional spiking neural network with adaptive coding for motor imagery classification," *Neurocomputing*, vol. 549, p. 126470, 2023.

[25] B. Singh and A. Kaur, "A comprehensive survey of regression based loss functions for deep neural networks," *arXiv:2211.02989*, 2022. [Online]. Available: https://arxiv.org/pdf/2211.02989

[26] S. Ghosh and S. Sahoo, "Loss functions and metrics in deep learning: A review," *ResearchGate*, 2023. [Online]. Available: https://www.researchgate.net/publication/372163006_Loss_Functions_and_Metrics_in_Deep_Learning_A_Review

[27] N. Skatchkovsky, H. Jang, and O. Simeone, "Spiking neural networks—part iii: Neuromorphic communications," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1746–1749, 2021.

[28] DataCamp, "What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained," https://www.datacamp.com/tutorial/

what-is-a-confusion-matrix-in-machine-learning, Nov. 2024, updated Nov 10, 2024; Accessed May 23, 2025.

[29] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford, UK: Oxford University Press, 2001.

[30] M. I. Skolnik, *Radar Handbook*, 3rd ed. New York, NY: McGraw-Hill, 2008.

[31] H. L. V. Trees, *Detection, Estimation, and Modulation Theory, Part I*. Hoboken, NJ: Wiley, 2001.

Acknowledgments

I want to sincerely thank Dr. Jacopo Pegoraro for his great support and guidance during my research. He was always patient and ready to answer my questions and helped me understand the complex theory behind this work.
Over the past few months, I have learned a lot, not just about implementing SNNs, but also about digital signal processing and creating realistic datasets. This experience has really improved my knowledge in the field.