Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2021-00

M.Sc. Thesis

Sound-recognition using Spiking Neural Networks

Randy Prozée

Abstract

The development of the Spiking Neural Network (SNN) offers great potential in combination with new types of event-based sensors, by exploiting the embedded temporal information. When combined with dedicated neuromorphic hardware it enables ultra-low power solutions, and local on-chip learning. This work implements and presents a viable architecture and training methodology to detect and classify audio data using Spiking Neural Networks.

The architecture consist of two core components: the first component is an auditory front-end which performs low level feature extraction. The second component is the SNN classifier supported by the spike encoder and decoder. The results show that the encoder has a major impact on the overall performance of the network. The temporal based network is trained with help of common training methods, both supervised and unsupervised. The performance of the network is validated under both clean and different levels of noisy conditions. The impact on classification performance is analyzed and compared with traditional non-spiking Artificial Neural Networks. This in terms of classification accuracy, estimate energy consumption and latency of inference.

The proposed architectures achieve a max accuracy of 97.0% under ideal conditions. This is comparable to other non-spiking artificial neural networks, which require significantly more energy for inference. The implementation demonstrates that the architecture is a viable solution for detecting and classifying audio data.

Sound-recognition using Spiking Neural Networks
My Subtitle

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Embedded Systems

by

Randy Prozée
born in Nieuwegein, The Netherlands

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Sound-recognition using Spiking Neural Networks"** by **Randy Prozée** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 26-02-2021

Chairman: _____

prof.dr.ir. T.G.R.M van Leuken

Advisor: _____

dr.ir. S.S. Kumar

Committee Members: _____

prof.dr.ir.z Z. Al-Ars

_____

dr. A. Zjajo

Abstract

The development of the Spiking Neural Network (SNN) offers great potential in combination with new types of event-based sensors, by exploiting the embedded temporal information. When combined with dedicated neuromorphic hardware it enables ultra-low power solutions, and local on-chip learning. This work implements and presents a viable architecture and training methodology to detect and classify audio data using Spiking Neural Networks.

The architecture consist of two core components: the first component is an auditory front-end which performs low level feature extraction. The second component is the SNN classifier supported by the spike encoder and decoder. The results show that the encoder has a major impact on the overall performance of the network. The temporal based network is trained with help of common training methods, both supervised and unsupervised. The performance of the network is validated under both clean and different levels of noisy conditions. The impact on classification performance is analyzed and compared with traditional non-spiking Artificial Neural Networks. This in terms of classification accuracy, estimate energy consumption and latency of inference.

The proposed architectures achieve a max accuracy of 97.0% under ideal conditions. This is comparable to other non-spiking artificial neural networks, which require significantly more energy for inference. The implementation demonstrates that the architecture is a viable solution for detecting and classifying audio data.

Acknowledgments

First, I would like to express my gratitude to dr.ir. T.G.R.M van Leuken for giving me the opportunity to work on this project. The opportunity to work on innovative technologies was inspiring and highly educational, as I gained a tremendous amount of knowledge.

I would like to thank my daily supervisor dr.ir. S.S. Kumar for the support and his continued availability for discussions and collaboration. Without his and dr. A. Zjajo guidance and feedback, I would not have been able to write this report.

To my colleagues at Innatera, CAS group members and fellow master students for creating a pleasant work environment. Highly appreciated at a time when contact was mostly virtual due to the global outbreak of COVID-19. Your support and fellowship pulled me through the tough moments.

The completion of my thesis was not possible without the support of my family and friends. Their continuous love and encouragement helped me through my entire academic journey.

To all, thank you!

Randy Prozée
Delft, The Netherlands
26-02-2021

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the deep learning revolution has become increasingly evident. Voice-activated personal assistants such as Sire and Alexa are influencing our daily lives. These breakthroughs are possible thanks to the use of artificial neural networks (ANN). These complex structures are often multiple layers deep, where intelligence is provided by learning techniques typically based on backpropagation. This supervised algorithm requires tremendous amounts of labelled training data to achieve an impressive classification accuracy, sometimes even exceeding human abilities. However, these advancements often consume large amounts of energy and thus limits its applicability. Spiking Neural Networks (SNNs) can be seen as next generation of Neural Networks, and hold the potential to overcome the energy obstacles. It performs brain-inspired information processing, passing and processing asynchronous spike potentials (binary signals), all highly parallel. Unlike traditional ANNs, Spiking Neural Networks incorporate temporal information into their operating model, meaning the concept of time. This neuron model is thus more biologically realistic than ANNs, carrying over one of most favorable characteristic of the brain, energy efficiency. When combined with purposely designed neuromorphic hardware, SNNs based solution can obtain Ultra-low power consumption, low latency for inference and potentially allow for local on device learning.

1.1 Problem statement

Automatic speech recognition (ASR) solutions have become prominent in our everyday life. An increasing number of smart devices have a voice interface, e.g. mobile devices and smart appliances. This surge in voice-controlled devices is made possible by recent advances in Neural Networks. Here complex network models are trained to perform speech recognition tasks. These large network structure require immense amount of computational operations, thus also increasing the power requirements. This obstacle has led to the utilization of cloud computing, where the processing load is transferred off device to external servers. However, this solution is far from ideal since sensitive user data is shared, leads to increased latency and requires a robust internet connection.

The goal of this work is to develop an algorithm/pipeline that enables on-device ASR using Spiking Neural Networks. The focus is on compact network architectures that allow for local on-device processing. Spiking Neural Network provide an attractive platform, especially for event-driven temporal information. Its favorable characteristic in terms of latency and power consumption could enable always-on speech recognition, which is highly desired. When viable, this solution could highlight the SNNs' potential and applicability to a broad market.

Research questions To guide the research, overarching research questions have been formulated as follows:

- How does the size of the Spiking Neural Network architecture affect overall performance, particularly classification accuracy?
- How do different quantities of background noise affect the classification accuracy?
- How does the Spiking Neural Network implementation compare to more traditional non-Spiking Neural Networks, in terms of classification accuracy, power consumption and latency of inference?

1.2 Thesis objective

The main objectives of this work are as follows:

- Develop a Spiking Neural Network architecture capable of classifying audio data with high accuracy.
- Enhance the network architecture to perform more robust classification under different levels of background noise.
- Research or develop an auditory front-end capable of encoding acoustic data into sparse spatio-temporal spike trains.

1.3 Contributions

The main contributions of this work are listed below:

- An efficient auditory front-end, inspired by the human auditory system, that results in low-level feature extraction.
- A baseline temporal-based Spiking Neural Network architecture that is highly capable of speech classification, which is inspired by the current state-of-the-art.
- An enhanced Spiking Neural Network architecture optimized for noise robustness, capable of performing under different levels of background noise.
- A software framework that utilizes a supervised learning method capable of training a temporal-based Spiking Neural Network.
- A tool capable of adding background noise to a clean speech signal, where the signal-to-noise ratio can be configured by the end user. It offers great control over the amount of noise subjected to the network, and allows for thorough analysis and verification on the impact of noise.
- An analysis of the robust speech classification network performance and the benefits compared to more traditional non-Spiking Neural Networks.
- A full system analysis that shows the proposed architecture can be a viable solution for robust speech recognition with Spiking Neural Networks.

1.4 Thesis outline

The thesis is structured as follows:

Chapter 2 provides background knowledge regarding Neural Networks. It explains the difference between typical Artificial Neural Networks and Spiking Neural Networks. The neuron and synaptic models on which the networks are build, the learning methods, simulation tools and a description of the Auditory front-end.

Chapter 3 contains an in depth description of the proposed network architecture. All the components of architecture are disclosed together with the enhancements made for improved robustness to noise.

Chapter 5 discussed the obtained results. Here, performance comparison are made of different sized networks, the gains of the enhancements and the impact of background noise.

Chapter 6 describes the overall outcome of this research. Based on the performance results the strengths and weaknesses of the architecture are uncovered.

Chapter 7 concludes the thesis and provides a list of possible future research topics to extend the current solution.

2 Background

In this chapter, background information is provided with regard to Spiking Neural Networks and neuromorphic hardware in general. It tries to answer the following questions. What is the definition of a Spiking Neural Network, and how to they compare to more common artificial neural networks? Which tools are available to simulate the network's behavior? What kind of encoding is required, so the input data can be processed by a Spiking Neural Network? How do biological systems encode and process audio information? Before going into more detail, a more in depth description of the use-case is given. This since the use-case plays a crucial part in the direction of this research.

2.1 Use-case description

Within the automotive industry, there is a need for on-board speech recognition solutions which can be deployed on embedded devices. This is especially relevant to enable vehicle access via voice (e.g. unlocking the car) from inside or outside the car. To support the vehicle access via voice, an array of microphones is mounted in critical locations.

The proposed solution, speech recognition with neuromorphic computing on embedded devices, has the potential to solve these issues. As described, the combination with Spiking Neural Networks exhibit favorable properties could meet the critical requirements. Other than the power requirements, other critical requirements are:

- **High accuracy**, since a large amount of miss-classification would make the solution unreliable.
- **Works in noisy environments** as for both the interior and exterior space background noise is expected.
- **Low latency** for inference, this because a slow response time would hurt the user experience.

Use-case examples
In a general sense, speech recognition systems utilized in the automotive industry aim to remove distractions and improve the user experience. The industry is exploring various modes for vehicle access, e.g. trigger action of the vehicle by voice from outside the car. In Figure 2.1 one of the important scenario is Illustrated, the two important voice triggered actions being;

- Lock/unlocking the vehicle
- Open/close tail gate (powered)

As described in future enhancements, ultra-low power speech recognition offers a great amount of new opportunities. This is certainly the case for interior space, where its use could be extended to the secondary controls of the vehicle. Voice activated secondary controls allow the user to easily operate the lighting, windows, wipers etc, while safely keeping both hands on the steering wheel.

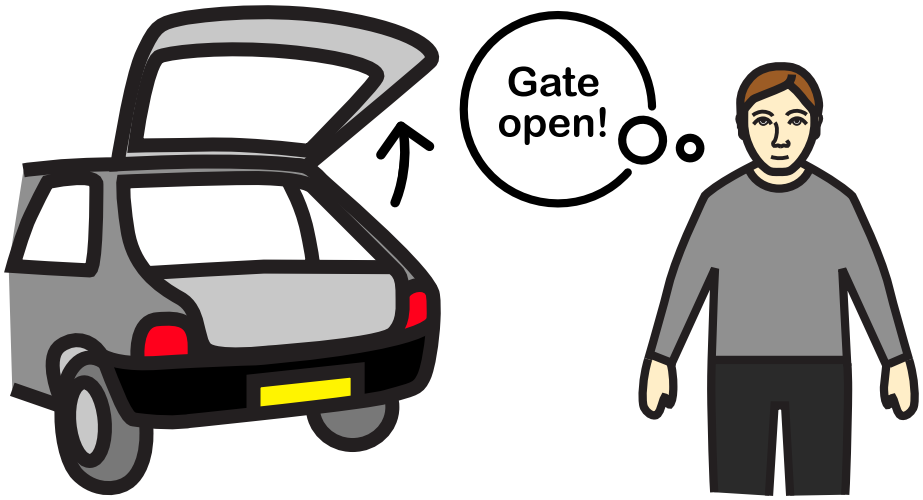


Figure 2.1: Visual representation of the voice-triggered tail gate action, powered tail gate open or close.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN), as the "neural" portion of the name implies, are compute models inspired by biological neural networks. The computational model was created by the neurophysiologists Warren McCulloch and the logician Walter Pits in 1943 [1]. At present, ANNs have gained significance in artificial intelligence with the advent of a training algorithm called backpropagation [2]. This technique enables artificial neural network to adjust its behavior when its predicted solution doesn't match with was expected. It enables researchers to create self-adjusting networks that can modify the connection strength between neurons, and thus increases the overall accuracy in networks. Unlike typical computer programs, which always execute according to the commanding of the programmer, it learns from examples and experiences not from predefined lines of commands. As a result, it enables computers to solve complex problems which are not know to humans more efficiently. For example, in the field of Computer Vision and Natural Language Processing.

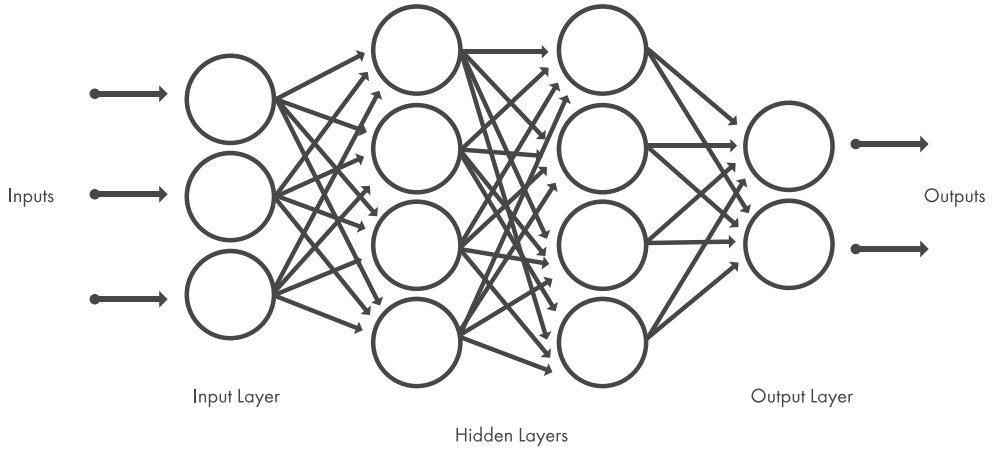


Figure 2.2: General visualization of an Artificial Neural Network.

As illustrated in Figure 2.2, the network consist of an input and output layer. In most cases, a hidden layer is applied to convert the input into a usable form for the output layer. The network is formed by a set of artificial neurons, which are also knows as nodes. The function the nodes perform can be different per layers, as they have a specific value which is transmitted to all connected nodes. Connected nodes compute their value by some non-linear function of the sum of its inputs. Weights control the connection strength between the nodes and can be increase or decrease. These weights are usually adapted during the learning stage, which is where the intelligence of an ANN originates.

2.3 Spiking Neural Networks

Compared to traditional ANNs, Spiking Neural Networks (SNNs) more strongly mimic the information processing in biological Neural Networks. Like the brain, neural activity is represented by so-called spike trains, which are sequences of event potentials (spikes). These events are processed by the neurons and communicates with other neurons via specialized connections called synapses. The neuron in a SNN only fires if the sum of integrated inputs reaches a threshold, unlike the bounded propagation cycle of typical multi-layer perceptron networks [3]. When a neuron elicits a spike, this potential will be relayed to all connected neurons, affecting the potential of these neurons. The timing of the arrival of each spike is of great importance, in order to propagate the signal. Unlike more classical artificial neural networks, the temporal placement (time) is thus incorporate into the model. Similar to the brain, it's capable of continuous event-driven information processing in massive parallel fashion. All these characteristics make SNNs extremely suited for efficient modeling of temporal data. The biggest advantage of Spiking Neural Networks is their theoretically ultra-low power consumption when combined with specialized neuromorphic hardware [4]. The ability to solve complex problems very power efficiently offers new opportunities. The creation, simulation and training methods available are outlined in the following subsections.

2.3.1 Biological Neurons

In the brain, a neuron (nerve cell) is fundamental for all communication in the nervous system. Each neuron is composed of the cell body, which includes the soma, large branching axons and smaller branching dendrites, see Figure 2.3. The soma is the main excitable cell that accumulates incoming event potentials received through the dendrites. Whenever a neuron elicits a spike, the axon relays the outgoing potential to other neurons through the synapse. All together these form the neural circuits which can perform complex functions. To exemplify, the human brain consists of billions of neurons with trillions of synaptic connections [5][6][7].

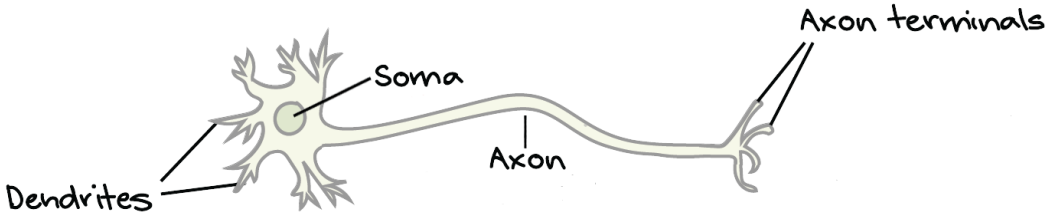


Figure 2.3: Anatomy of a neuron, highlighting the three main components, the soma, dendrites and axon.

The process of exchanging information between transmitting neurons can be outline in three main phases. Initially, the transmitting neuron generates an electrical pulse also known as an action potential or spike. This potential is released from the cell body and travels along the axon to the terminal endings. The dendrite of the receiving neuron is connected to a branch of the axon terminals via a synapse. This connection allows neurons to interact and transmit information from one to the other. This process only starts when a neuron receives sufficient stimuli, from one or multiple sources, as the incoming electrical energy is accumulated raising the membrane potential at the neuron's cell body. When sufficient energy is accumulated in a small time-interval, a certain threshold is exceeded, causing the neuron to produce its own action potential, or outgoing spike. This and the following phases of the neuron's behavior are illustrated in Figure 2.4. The action potential will be relayed to all other connected neurons. The accumulated energy in the transmitting neurons cell body will reset to its resting potential and the neuron will enter a refractory period. In this temporary state the neuron will not respond to incoming stimuli nor will it produce new spikes. In case of failed initiation (meaning insufficient stimulation of the neuron), the accumulated energy in the neuron cell body will slowly decrease until the resting state, since the cell membrane isn't a perfect insulator.

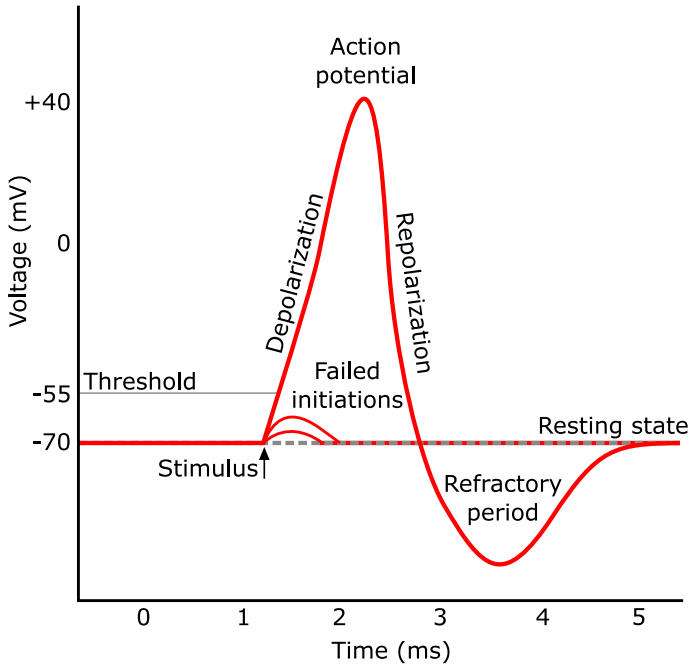


Figure 2.4: Illustration of an action potential, showing the various phases that occurs as stimuli effect the neuron's cell body.

2.3.2 Neuron Model

To exploit the computational potential of a neuron, a model is needed to recreate or simulate its behavior. The most famous being the biologically-inspired Hodgkin–Huxley model proposed in 1952 [8]. This comprehensive model describes the behavior of the biological neuron in great detail, included the generation, diffusion and accumulation of action potentials. The model is derived from biological experiments, by measuring actual cerebral activity. The electrical characteristic of the neuron are described by a set of nonlinear differential equations, shown in [8]. While this model represents a neuron with high biological plausibility, its implementation cost have shown to be prohibitive [9]. It did, however, lay the groundwork and initiate the development of many other less complex neuron models. Driven by efficient implementation cost the simplistic integrate-and-fire model emerged [10]. In this model neurons are viewed as integrators, which elicits an output spike only when its accumulated potential crosses the set threshold. Following the release of the potential, the neuron will enter its resting state. Due to the simplicity of the model, basic circuit consisting of a capacitor (C) can model this behavior. As highlighted by Equation 2.1, the neuron model is simply the derivative of the self capacitance of a conductor. Albeit its simplistic nature, the model is still wide used for analyzing neural network behavior.

$$I(t) = C_m \frac{dV_m(t)}{dt} \tag{2.1}$$

The leaky-integrate-and-fire (LIF) model is an extension of the previously described integrate-and-fire model. The addition of the leak term models the behaviour of the membrane potential more closely, as it's not the perfect insulator. It solves the memory problem of Equation 2.1, where the accumulated potential is not discharged over time. This added complexity can be easily implemented in hardware by the addition of a discharge resistor (R), allowing the membrane potential to return to its resting state. This extended model shown in Equation 2.2.

$$I(t) - \frac{V_m(t)}{R_m} = C_m \frac{dV_m(t)}{dt} \tag{2.2}$$

Here R_m is the membrane resistance, modeling the discharge characteristic of the membrane potential. This work utilizes the LIF model due to the hardware driven nature of the use-case. This level of abstraction is most suitable for the purpose of this work, since the efficiency and the implementation cost is significantly more important than the biological plausibility of the neuron model.

2.3.3 Synapse

The synaptic connections permit a neuron to pass signals to other neurons. These large structures are needed to form a network capable of handling complex tasks. An action potential is relayed from the cell body through the axon, which branches into multiple terminal ends. Each terminal end making various connections with the dendrites of receiving neurons via a synapse. Likewise, a single neuron may receive action potentials from numerous neurons through its dendrites. The exact arrival times of these action potential determine if a subsequent potential is generated. Figure 2.5 depicts the described synaptic connection.

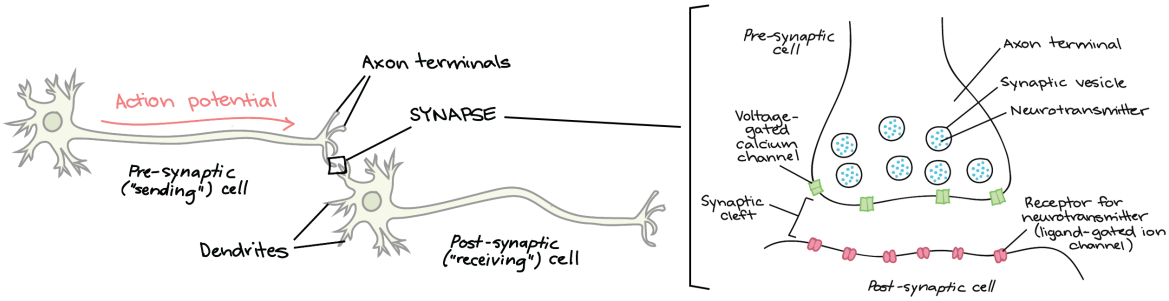


Figure 2.5: Illustration of the synaptic connection between two neurons and to the right a more in-depth view of the synapse.

In simple terms, the axon terminals of a sending neuron contains neurotransmitter molecules. In case of an action potential, these neurotransmitters are released into the synaptic cleft, the small gap between the pre- and postsynaptic neuron. The molecules will diffuse across the synaptic cleft on the postsynaptic cell and forms a chemical transmission. It enables the receiving neuron to generate its own postsynaptic events, which in turn will serve as the stimuli for the next receiving neuron. These synaptic connects are a critical component of neural networks yet, the biological model of the synapses is very complex. A simplified model is required in order to realize a cost-efficient implementation. This is achieved by the use a current-based synaptic model for both the simulator and neuromorphic hardware platform. The model adds a pre-set amount of current on the postsynaptic side, only in case of a presynaptic event. The specific amount of added current is defined by variable called a weight. By changing the synaptic weight between neurons, the behavior of the network can be altered and allows for networks with complex behavior.

2.4 Information encoding

Within Spiking Neural Networks, information is represented and communicated by spikes. These binary events result in a sequence of spikes, called spike trains, containing the information based on the encoding scheme applied. The encoding scheme used to encode external information plays a critical part in the functioning and performance of the network. Neural information coding are still actively research [11], since there is no universal solution has been found. Therefore, numerous coding options are

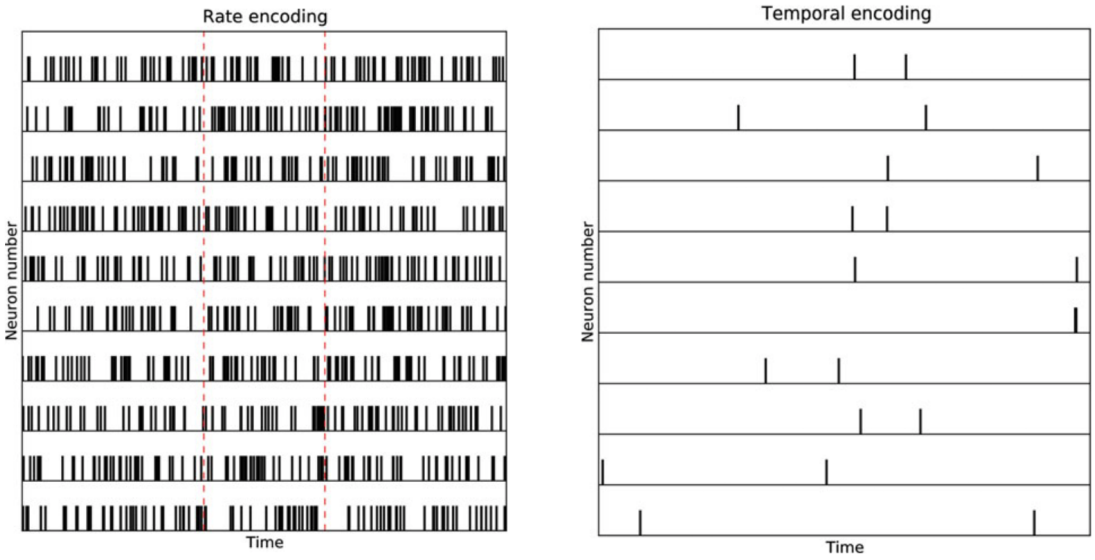


Figure 2.6: Rate encoding versus temporal encoding of information. Demonstrating the sparse and reduced frequency of the temporal spike trains.

available. These encoding options can be split into two main categories, namely rate encoding and temporal encoding. The encoding type used in the networks depends on the application and the training method used. Specific for this use-case, where temporal audio information needs to be processed, a temporal encoding methods seems the obvious choice. This since temporal encoding schemes fully exploit the acoustic temporal information leading to more sparse spike trains. A more detail description, including both the benefits and cons, will be given in Section 2.4.1 & 2.4.2.

2.4.1 Rate encoding

With Rate-based information encoding the frequency or rate of the spikes depends on the intensity of the input stimuli. As the intensity increases, the neurons firing rate will increase accordingly. It assumes that all stimuli information is contained in the firing rate of the neuron, thus neglected any temporal information possibly encoded by the timing of the spikes. This form of encoding is often used static applications, like [12], where pixel-based inputs with no temporal information are processed. Furthermore, recent experimental studies have provided evidence that a straightforward firing rate concept may be too simplistic to describe brain activity [13]. Another drawback of rate encoding is the large amount of spikes needed to encode the signal. When the input value approaches the maximum intensity value, the output will also have spike frequency close to its maximum. This large amount of spikes will negatively impact the power-efficiency of the network.

2.4.2 Temporal encoding

In contrast to spike rate encoding, where information is encoded based on the firing rate, temporal encoding is based on the precise timing of the spikes. Due to its sparsity of resulting spike trains, as is illustrated in Figure 2.6, each spike and its exact timing is of great importance. A wide range of temporal encoding strategies exist, with the disadvantage that they increase the complexity of training the network. As is demonstrated by the Tempotron learning rule, see Section 2.5, as its only capable of training single network layer at a time. Nevertheless, temporal encoding schemes greatly reduce the amount of spikes processed by the network, as each spike contains more information [14]. This will positively impact the power-efficiency as each spike increases the power requirements. A comparison between the spike trains obtained by rate encoding and temporal encoding is shown in Figure 2.6.

2.5 Training methods

Spiking Neural Network have the potential to outperform current Neural Network with regard to latency and power efficiency per inference. Research has shown [15] that SNNs theoretically can obtain the same level of computational complexity as traditional ANNs. However, in practice SNNs have often not reach the same level of performance. The theoretical potential is often hampered by the absence of powerful training methods that allow for deep SNN structures. This obstacle is caused by non-differentiable spikes used for information processing, which is fundamental for using error backpropagation techniques. Training methods based on this technique enables deep ANN structures, modeling complex behaviors. Numerous solutions have been proposed that workaround the obstacle of non-differentiability. Often inspiration is taken from unsupervised biological learning processes. Available training options for SNNs can be divided into three classes: unsupervised learning, indirect and direct supervised learning.

Unsupervised learning methods are often inspired by biology. A popular method for unsupervised learning is Spike Timing Dependent Plasticity (STDP) [16]. Here the weight are adjusted based on the correlations between pre- and postsynaptic spike events. This form of Hebbian learning [17][18] is often summarized as "Cells that fire together wire together". For interested readers, more detailed information can be found in [19][20][21]. Achieving high accuracies with training methods that employ Hebbian learning is often challenging and time-consuming without global supervision.

Indirect supervised learning is a more recent approach, where different data representation between training and processing are used. Within this process a traditional ANN model is trained and translated into a SNN equivalent. The intensity values used in the ANN computations are converted into firing rates for the spiking neural model [22][23]. The mapping from an ANN to SNN is not without drawbacks, given that there may be a performance loss for the SNN. The resulting rate-based SNN is often applied to static classification tasks where the loss of temporal details is less relevant (see Section 2.4). However, in this work temporal acoustic information is processed, any loss in temporal information will thus negatively impact the overall performance. In addition, the temporal benefits of the SNN will not be fully exploited. If anything,

it would be a missed opportunity to highlight SNN's favorable temporal event-driven characteristics.

Numerous direct supervised learning methods have been proposed, ranging from fairly simple to extremely complex learning rule. A popular lightweight option is the Tempotron [24] learning rule. This powerful learning rule is directly applied to spatio-temporal encoded spike trains. Like other learning rules that operate directly on the spike trains, limiting its training capabilities to single network layers. It is therefore very difficult to train deep SNN structure, as it's not applicable to hidden layers. This learning is discussed in more detail in Chapter 4 due to its extensive usage. Other more capable options include, Spikeprop [25] and ReSuMe [26]. Both workaround the obstacle of non-differentiability and thus are capable of training hidden layers. However, the increased capabilities come at the expense of added complexity. Thus making in more complex to implement, test and verify its functionality.

2.6 Spiking Neural Network simulations

The neuromorphic hardware platform that implements the leaky integrate-and-fire model is currently still under active development. Since this hardware platform is currently unavailable, experiments can only be performed by simulating the behavior of the Spiking Neural Network. Initial experience, with creating and testing Spiking Neural Network, was gained with the widely used Brain2 simulator [27]. This simulator is capable of testing a broad range of neuron and synaptic models which are defined by a set of differential equations. For example, the Hodgkin and Huxley, Izhikevich and the leaky integrate-and-fire models can easily be implemented with Brain2. Due to its popularity, an excessive amount of documentation and examples is available.

In the more advanced stage of the project the decision was made to use the proprietary in-house simulator. The main advantage of this simulator is its efficiency and speed in comparison with Brian2. Furthermore, it's fully applicable to the future neuromorphic hardware implementation currently in development. This is of great importance due to the specific use-case of the project. Here the ultimate goal is to demonstrate the functionality of both the network architecture and neuromorphic hardware platform itself.

All described results are obtained with the in-house simulator, which tries to meet the specific requirements and limitations of the intended hardware platform.

2.7 Auditory front-end

The auditory front-end is an important component of any speech recognition system. Its task is to decompose the incoming audio stream into distinct features. These features are critical for the performance of the subsequent classifier, as low-quality features can become the limiting factor. It is therefore important to provide qualitative acoustic features to ensure the success of this work. To achieve the objective, it is important to seek inspiration from biology, as evolution resulted in effective and efficient auditory

systems. E.g. the human auditory system performs extremely well for speech recognition, as so, it has served as the basis for most speech recognition systems. In simplified terms, the auditory system can be seen as a frequency analyzer [28, p. 289]. These actions are mostly handle by the cochlea, which contains the sensory organ of hearing. Its functionality can be emulated by non-linearly scaled filter banks which mimic the logarithmic filter behavior of the cochlea. This proven form of frequency decomposition is commonly used in both the ANN [29] and SNN [30] [31] domain. The aim here is to capture the spectral and temporal features by framing the output of the filter bank, resulting in a two-dimensional spectrogram. The spectrogram captures the frequency spectrum of the incoming autistic signal, as the signal varies with time. Next, all key acoustic feature embedded in the spectrogram need to be encoded into temporal spike trains.

A potentially more energy-efficient alternative is to encode the output of the filter bank directly. However, this solution is very experimental and high performance is not guaranteed. Because of the complexity and time constraints of this work, it was decided to focus on proven methods. If time permits, other solutions will be explored.

3 Architectural design

This chapter offers an overview and in-depth analysis of the architectural design. First, a global outline is provided, after which each component is examined in greater depth. The architecture consist of four main parts:

- Auditory front-end
- Spike encoding
- Spiking Neural Network
- Spike decoder

Each of these components plays a critical role in the functioning and performance of the architecture. The reasoning and exact function of each component is elaborated in the following sections.

3.1 Architecture

A high-level overview of the proposed architecture is depicted in Figure 3.1. The sensory audio data received is processed by the auditory front-end which mimics the human auditory system. It decomposes the sensory data in a set of frequency components used for low-level feature extraction. The temporal information of these frequency components are captured by splitting the signal into short frames. For each frame the spectral energy is calculated, thus obtaining the strength of each frequency component over time. The visual representation of the resulting output is called a spectrogram, see Section 3.1.1 for a more detailed description. In the next stage, the encoder converts the information embedded in the spectrogram into sparse temporal spike trains. This encoding step is necessary so that the information can be processed by the Spiking Neural Network, which acts as a classifier. In the final stage, the output of the SNN classifier is interpreted by the decoder and the corresponding label is selected.

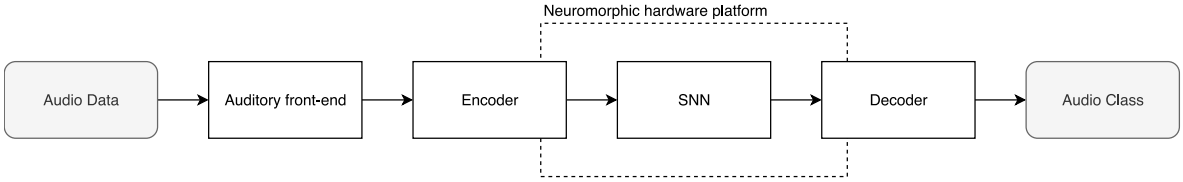


Figure 3.1: Proposed architectural overview. The architecture consist of four main parts, the auditory front-end, encoder, SNN classifier and decoder.

3.1.1 Auditory front-end

Automatic speech recognition (ASR) systems are used extensively in, e.g. mobile phones, and has guided the design of many useful speech processing algorithms. State-of-the-art speech recognition systems [32] [33] rely on fixed, handcrafted features such as mel-filterbanks to preprocess the audio waveform. The constantly changing audio signal passes through a cochlear filter bank, decomposing the signal into multiple components, each carrying a single frequency sub-band of the original signal. The cochlear filter bank is designed to emulate the human auditory system. The nonlinear perception of sound is replicated through the use of the Mel-Scale, it being more discriminative in the lower frequency ranges. The filter bank parameters are in Mel (m) and Hertz (f) are calculated with help of the subsequent equations:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{3.1}$$

$$f = 700 \left(10^{m/2595} - 1\right) \tag{3.2}$$

The filter bank consist of a set of triangular filter, with a response of 1 at each mel-scaled centre frequency. An overview of the exact response of each filter is illustrated in Figure 3.2. The use of Mel-Scale offers great flexibility, as the number of filter channels and the active frequency range in Hz are given as parameters. This flexibility is of great value since it doesn't limit the input audio characteristics. In contrast to fixed predefined filter parameters, which are purposely designed for a given audio signal. Based on performance results form other research [32] [33], an 20-channel filter bank is applied as these settings have shown strong results for similar applications. The exact filter parameter can be found in Appendix A.1. The filter bank is implemented by a set of analog band-pass filters, omitting the needs for the more computational expensive Fast-Fourier transform. This to minimize the energy requirements of the architecture. In case a more flexible generalizable architecture is desired, the use of Fast-Fourier transform is recommended.

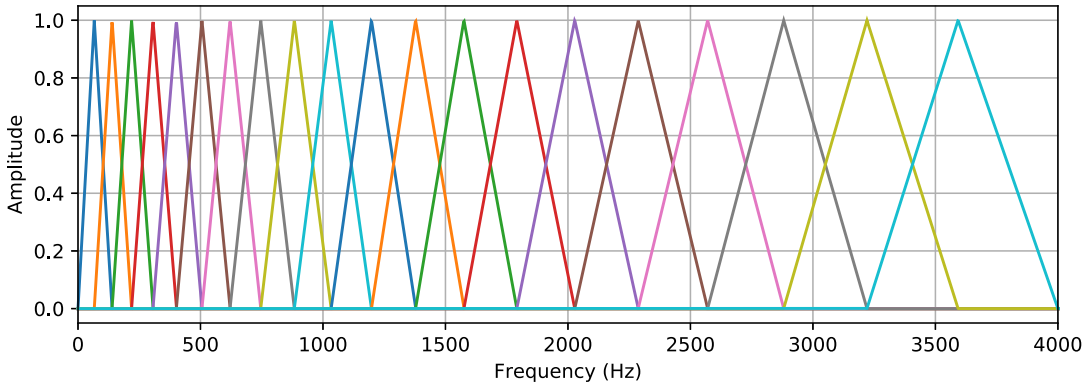


Figure 3.2: Example of a 20 channel filter bank on a Mel-Scale with a frequency range of 0 to 4 kHz.

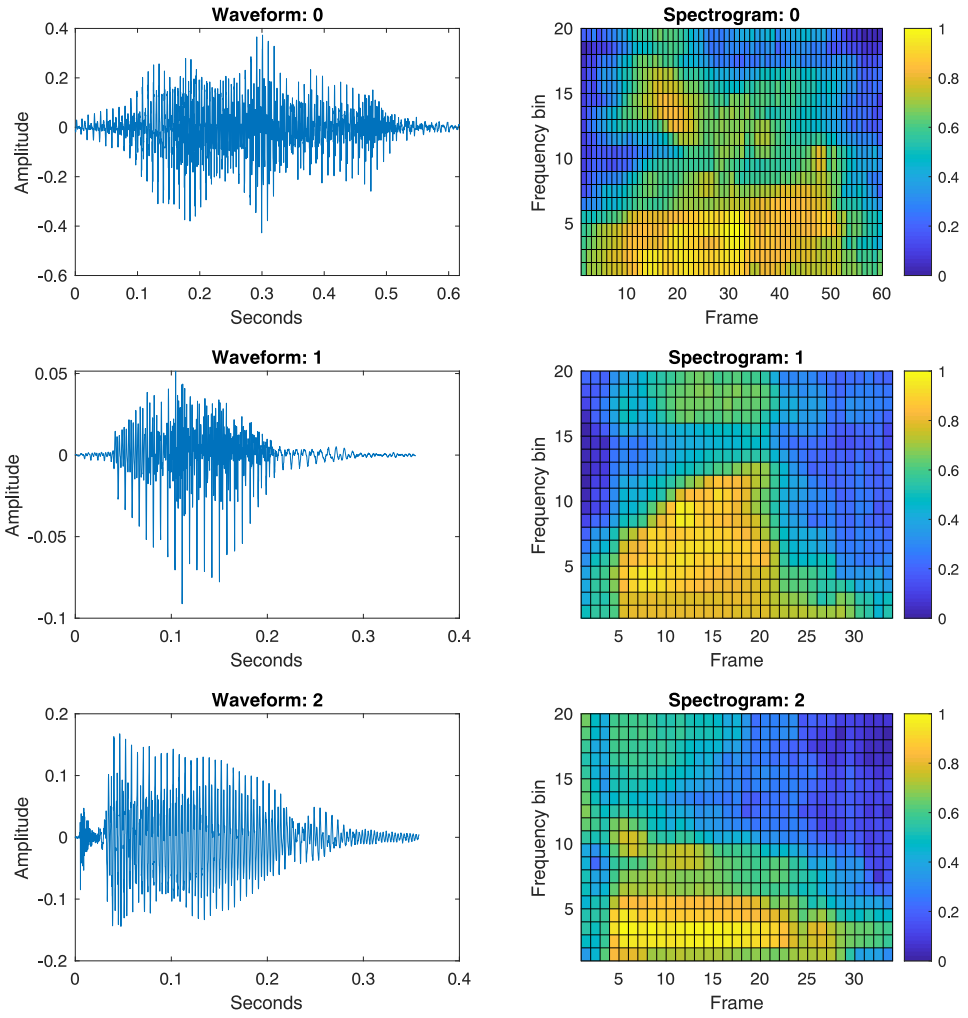


Figure 3.3: An illustration of audio waveforms and there corresponding spectrogram. The audio samples contain the utterance of spoken digits 0 to 2.

The analog filters are implemented following the Butterworth design. This filter design is chosen for its flat as possible frequency response in the pass-band. These characteristics are desirable as the passed frequency components at the output of the filter will be left mostly unaltered. The 20-channel signal output of the filter bank is framed into 20ms windows. In general a windows sizes in the range of 20-40ms is applied for speech applications. A much shorter windows size will lead to an unreliable spectral estimate due to insufficient samples, if longer the signal changes within the frame are not captured. Furthermore, a stride size of 10ms is applied resulting in a typical 50 percent overlap of the frames. After slicing the signal into frames, a Hamming window is applied to reduce the spectral leakage due to windowing.

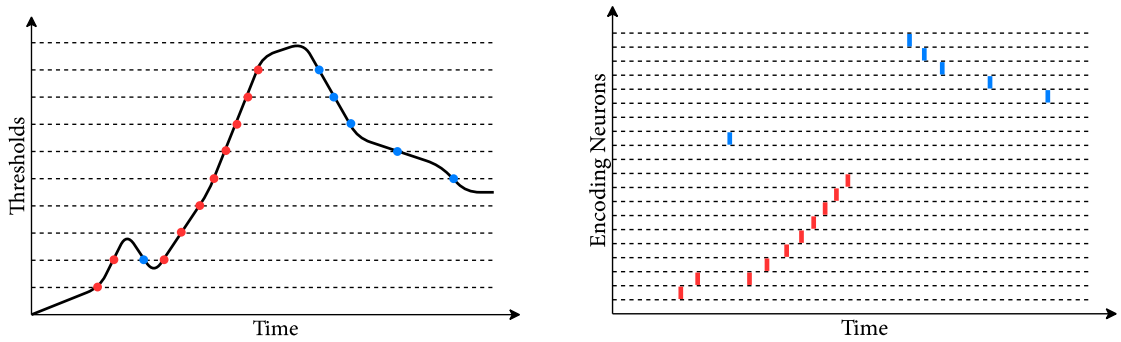
In the following step the spectral energy of each frame is calculated. This step mimics the human cochlea (see Section 2.7) which identifies the frequencies present in the incoming sound. The spectral energy ($logE$) is computed following Equation 3.3, where N is the frame length and F is the frame content. The logarithmic part of the Equation 3.3 emulates the non-linearity of the biological auditory system.

$$logE = ln\left(\sum_{i=1}^{N} F(i)^2\right) \tag{3.3}$$

The output of the auditory front-end is a two-dimensional spectrogram which contains spatio-temporal information. A number of example spectrograms are shown in Figure 3.3, the 20 rows represent the frequency bins used to decompose the signal. The unique features embedded in each waveform are captured and highlighted by each spectrogram. These features play a cricial role in speech recognition, and enable the subsequent stages to classify the incoming audio data. Note that each spectrogram is normalized in preparation for the following encoding stage.

3.1.2 Spike encoding

The audio information processed by the Spiking Neural Network has a strong temporal nature. Furthermore, sparse spike trains are desirable for a greater power-efficiency and fast inference. Due to these characteristics a form of temporal encoding called population threshold algorithm (PTA) is utilized in this work. The population contains a set of encoding neurons, each with a unique set threshold. In this case each threshold is linearly spaced within the normalized range of each spectrogram, thus between 0 and 1. Each time the incoming signal crosses a threshold, a peak is generated, thus translating each spectrogram into the spiking domain. The results shown in [32] highlights the efficient performance of this encoding methods. Compared to other common encoding options, this method achieves the lowest average spike rate, while performance is similar. In some cases even outperforming them, as is the case for latency and phase coding.



(a) (b)

Figure 3.4: Population threshold encoding: (a) the varying signal crossing the set of linearly spaced thresholds, (b) the resulting temporal spike trains corresponding to each crossing. The red labeled spike correspond to crossing from underneath and the blue labeled refer to crossing from above.

Threshold-based encoding models generate a spike only when the incoming signal crosses a certain threshold. Instead of using a single neuron to encode the signal, multiple neurons (a population) are used to encode the signal, each having their own specific threshold value. These values must be in range of the input signal as they define the selectivity of the neuron population. As illustrated in Figure 3.4, a distinction is made between crossing from underneath and above. This approach omits the need of inhibitory (negative) spike, as both onset and offset spike trains are fed into the SNN. It doubles the amount spike trains required, in this example 10 thresholds leads to 20 spike trains. The varying signal in Figure 3.4a crosses a set of thresholds, depicted by the dotted lines. For each crossing a spike is generated, resulting in temporal encoded spike trains. The simplified pseudocode for this type of population threshold encoding is depicted below by Algorithm 1.

The described method is used to encode the spectrogram in a set of sparse temporal spike trains. The conversion from a real spectrogram to spikes is illustrated in Figure 3.5. In this case 15 uniformly distributed thresholds values, within the normalized range of 0 to 1, have been applied to each spectrogram. The resulting 600 spike trains contain the speech features in a more abstract from. In the next stage the spike encoded information stream is processed by the Spiking Neural Network.

Algorithm 1: Population Threshold Encoding algorithm pseudo-code

Data: *signal* $\rightarrow$ a T-length vector containing the signal values
thresholds $\rightarrow$ a H-length vector containing the thresholds
Result: *onset_spikes* $\rightarrow$ a T*H-sized matrix containing temporal onset spikes
offset_spikes $\rightarrow$ a T*H-sized matrix containing temporal offset spikes

```
memory(1 ... H) = false;
onset_spikes = zeros;
offset_spikes = zeros;
foreach T → t do
    foreach H → h do
        if memory = false AND signal(t) > thresholds(h) then
            onset_spikes(t, h) = 1;
            memory = true;
        else if memory = true AND signal(t) < thresholds(h) then
            offset_spikes(t, h) = 1;
            memory = false;
```

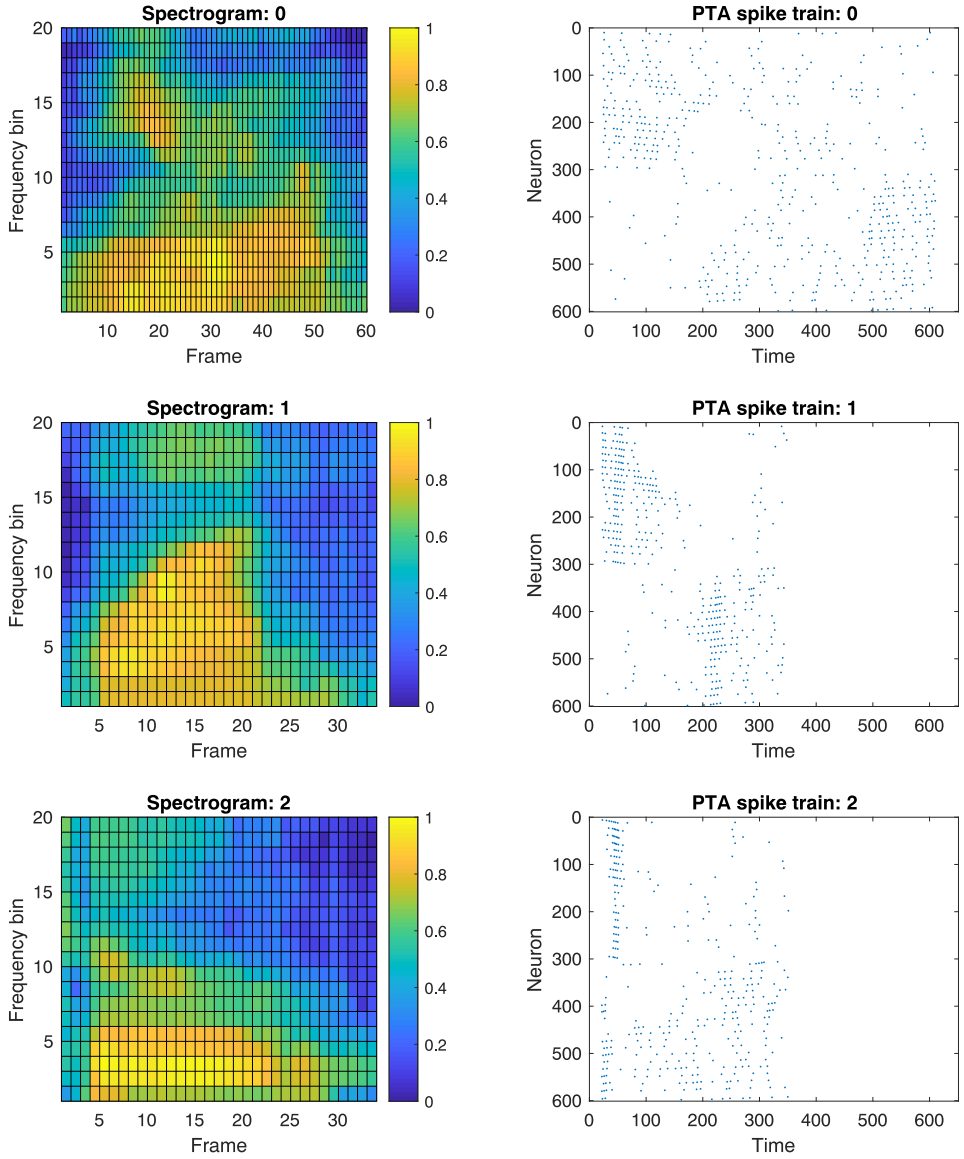


Figure 3.5: Overview of audio spectrogram and the corresponding spike trains obtained by population threshold algorithm (PTA) encoding. For this example 15 uniformly distributed threshold values are used, resulting in 600 sparse temporal spike trains. Like in Figure 3.3 the audio samples contain the utterance of spoken digits 0 to 2.

3.1.3 Spiking Neural Network

The architectural design of the Spiking Neural Network is strongly affected by external factors. In this particular case, the use of a temporal-based network has implications on the training options. Training methods capable of training these type of network are often limited to single network layers, or become extremely complex when capable of training multiple layers. At the start of the project, the simulation toolbox did not

include a training option for temporal-based network. This obstacle has been overcome by the implementation of Tempotron learning rule. More about this relatively simple supervised learning method can be found in Section 2.5. The use of this learning method adds a constraint, as it is only capable of training single network layers. Due to this limitation, a network architecture that is only a single layer deep is implemented. A fully connected network structure is implemented containing only an input and output layer, thus no hidden layers. An illustration of the network architecture is shown in Figure 3.6. The size of the input layer is fully dependent on the number of spike trains presented to the network. For example, a PTA with 15 threshold values requires an input layer size of $(2 \cdot 15\ thresholds \cdot 20\ bins =)$ 600. The output of the network has a fixed size which corresponds to the number of classes used. Different sized input layers have been explored and the performance impact has been studied. The observations made, like classification accuracy and robustness to noise, are described in Chapter 5.

The exact parameters for neuron simulation can be found in A.2. These parameters are based on the neuromorphic hardware platform itself, thus comparable hardware classification accuracies are expected.

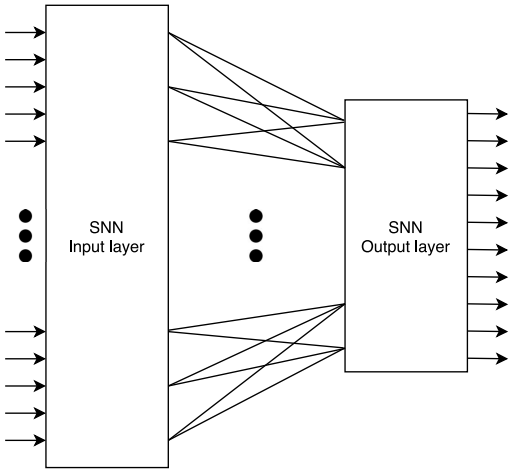


Figure 3.6: Architectural overview of the SNN classifier. The size of the input layer is fully dependent on the number of spike trains presented to the network. The output layer has a fixed size, one for each output class.

3.1.4 Spike decoder

The raw SNN output is still in spike format, with help a decoder these spike trains are interpreted. Within the simulator toolbox a variety of decoder options are available. The optimal decoder was found to be the First-time-to-spike method, due to the sparse output of the network caused by the Tempotron learning rule. The frequency-based decoders suffer from ambiguous classification due to the low amount of output spike. For example, in case two output both only have a single spike output, the spike frequency is equal, leading to misclassification as the decoder cannot distinguish them.

As demonstrated in Figure 3.7, the First-time-to-spoke decoder selects the output with the earliest response. In the case Figure 3.7 label 1 is selected as its output had the

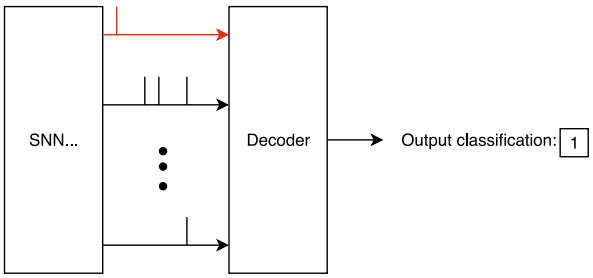


Figure 3.7: An example scenario in which the First-time-to-spike decoder is applied to a set of spike trains.

fastest response. In contrast to the frequency-based decoders, this method is capable of classifying spike outputs which contain on single spikes. The only drawback with this approach is that it's receptive to noise. When an early unwanted output spike is generated because of noise, a misclassification will occur. The frequency-based method, which often requires much more spiky output, is potentially less receptive to unwanted spikes. This since, unwanted spikes are less likely have a significant impact on the spike frequency and thus its classification.

3.2 Enhancements

In this section improvements made to the architecture will be discussed. This in terms of classification performance, network size, number of spikes, and noise robustness. The impact of the improvements are described in Chapter 5.

3.2.1 SOM layer encoding

With this enhancement, an unsupervised self-organizing map (SOM) is used for representing the frequency content captured by the auditory front-end. The spike encoder, discussed in Section 3.1.2, is replaced by the unsupervised SOM which generates an effective and sparse mid-level representation of the embedded features. Other research [34] made the observation that existing SNN temporal training rules cannot effectively discriminate latency or population encoded filter banks. Therefore, an SOM based solution is proposed to form low-level feature representation of each spectrogram. In Figure 3.8 the architectural chances are depicted. The red color highlights the trainable components of each architecture. The enhanced architecture is a multi-layer network where the SNN input layer forms a hidden layer. Self-organizing map is capable of dimensionality reduction but also reduces the effects of noise (variance) and redundancy (highly correlated variables). These characteristics combined with low-level spectral features extraction of each frame, allow the SNN classifier to be more robust to background noise, as is demonstrated in Chapter 5.

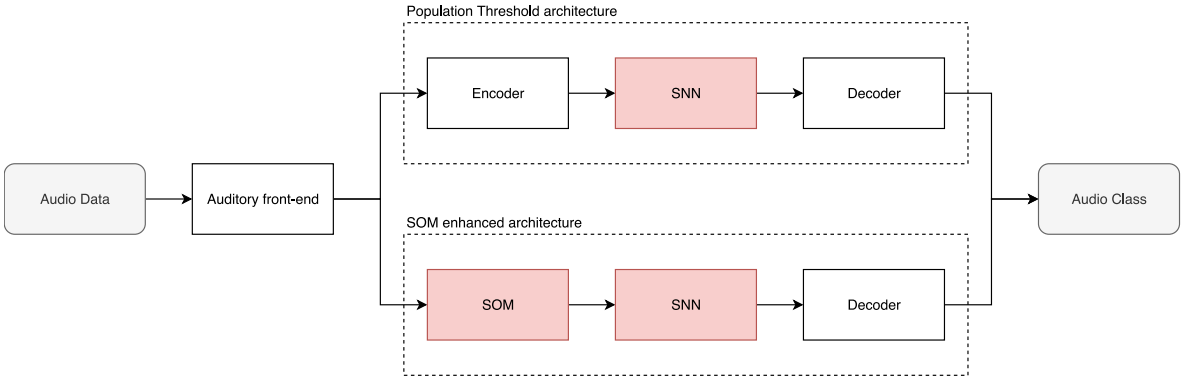


Figure 3.8: Architectural overview of the initial population Threshold encoded architecture and the SOM enhanced architecture. The latter utilizing the SOM for mid-level feature representation of each acoustic spectrogram.

The unsupervised SOM is implemented with help of the MATLAB Neural Network Toolbox. The SOM maps the input vector onto a dimensional space, where the weights in the feature map can be seen as coordinates. This spatial translation presumes that closely connected neurons and inputs share similar properties. Figure 3.9 demonstrates a small fully connected SOM network of $(4 \cdot 4 =)$ 16 neurons. Unlike typical artificial networks which apply error-correction learning (e.g. backpropagation), the SOM uses competitive learning. A given frame of the spectrogram is presented to the SOM, the similarity compared to each weight vector in the feature map is calculated with help of the Euclidean distance. The neuron which most closely resembles the input vector is selected, this neuron is called the Best Matching-Unit (BMU). The weight vector of the BMU and the neurons in its proximity are adapted to more closely represent the input vector. The scale of this adjustment is based and on the distance to the BMU, and also decreases per epoch. The BMU within the SOM are activated over time as the spatio-temporal input data is encoded into tonotopically organized feature maps, where each sparsely activated BMU represent a spiking neuron. The spikes triggered over the duration of an audio sample result into spatio-temporal spike trains, which are processed by the SNN and are classified as one of the audio classes.

In Figure 3.10 a comparison is made between the spike trains obtained with the population Threshold algorithm (PTA), and the output of the SOM. In case of the SOM implementation, its output clearly shows the low-level feature extraction of the audio data. These low-level features are not as pronounced with the Threshold encoded signal. They are encoded in a more abstract form and require more spikes to represent the same input data. This is an important side note, as each elicit spike will increase the power consumption of the network.

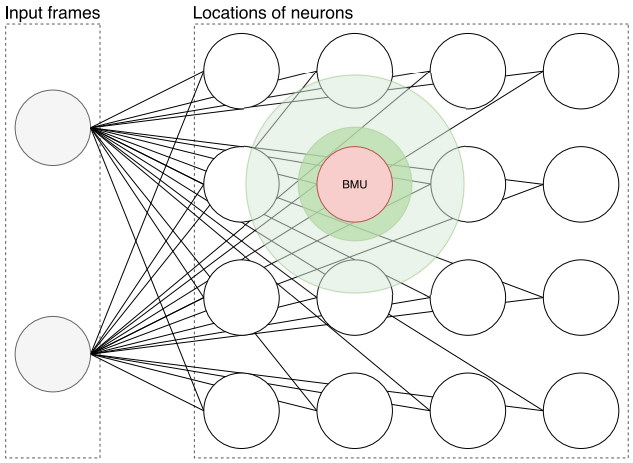


Figure 3.9: A high-level representation self-organizing map (SOM). The neuron most similar to the input vector is identified as the Best Matching-Unit (BMU). The weight vectors for each neuron in its proximity, highlighted by the green zone, are adapted to the input vector.

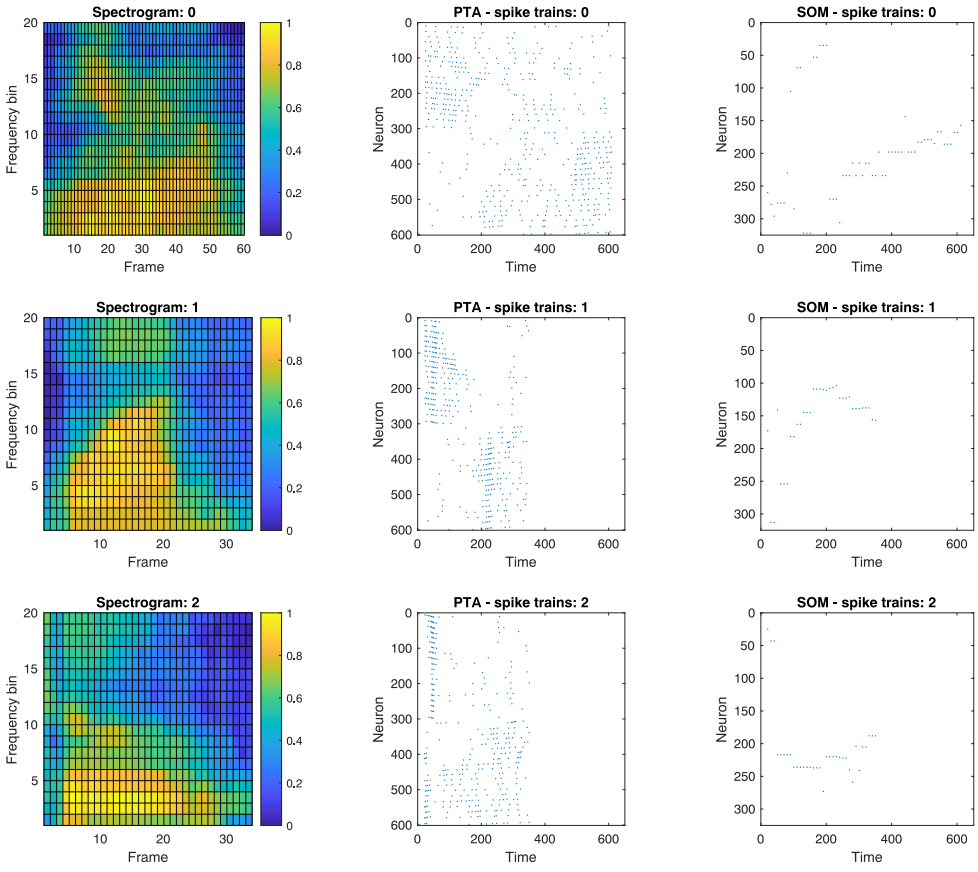


Figure 3.10: A visual comparison between the resulting spike trains for Population Threshold algorithm (PTA) and self-organizing map (SOM). The resulting low-level features by the SOM are distinct, and requires less spike to represent the same data.

3.2.2 Dedicated noise class

An important characteristic of any robust ASR system is its capability to distinguish speech from background noise. As shown in Chapter 5, the proposed network architecture performs poorly in tests where only background noise is presented. This is a result of the binary classification applied to the output of the network, where each class has its own dedicated output neuron. Due to the limited depth of the network, no hidden layers, there is a high likelihood that a noise fragment will elicit an output-spike. Causing a high amount of misclassifications where the noise fragments are identified as speech classes.

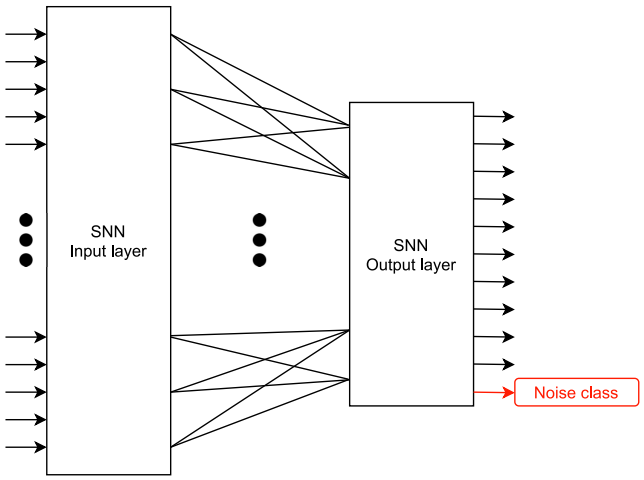


Figure 3.11: Architectural overview of the Spiking Neural Network with the added noise class, depicted in red. The output layer has a fixed size, one for each output class, plus the one dedicated to noise.

In the ideal case, no output spikes occur when only noise is presented to the network. This behavior is difficult to recreate due to the limited depth of the network. A similar type of behaviour can be obtained by adding a dedicated noise class, and thus output, to the network architecture. An illustration of the addition of a dedicated noise class is depicted in Figure 3.11. This enhancement allows the output-spikes, as a result of the noise fragments, to be directed towards a dedicated output neuron. This architectural enhancement, in combination with added noise only samples, dramatically improves the networks capabilities to distinguish speech from noise, as is discussed in more detail in Chapter 5.

3.3 Non-spiking neural networks

For the purpose of demonstrating the performance of the proposed SNN architectures, their performance is benchmarked against non-spiking ANN solutions. These network models are process the same spectrogram data as the proposed SNN implementation. Three common network architectures have been selected, namely the Convolutional Neural Network (CNN), the Recurrent Neural Network (RNN) and the Long Short-Term Memory (LSTM) [35]. All network models are implemented using the Keras

open-source library for artificial neural networks [36]. This selection of neural networks covers a broad variety of use-cases. For example, the CNN is often used for static image processing [37], whereas the RNN are used for temporal processing [38] and the LSTM are highly suited to Natural Language Processing (NLP). The exact network architectures used are shown in Table 3.1. For a fair comparison, all ANN architectures are compact in design, as this is the case for the proposed SNN architecture. This restriction impacts the overall performance and power consumption of each implementation and therefore leads to a more honest assessment. It's worth noting that limited time is devoted to optimization the ANN networks, e.g. search for the most optimal number of parameters, as the exploration of the SNN networks was already time-consuming.

Network model	CNN	RNN	LSTM
Network layers	Conv2D layer 32*	LSTM layer 30 (Recurrent)	LSTM layer 30
	Conv2D layer 16*	Dropout layer 0.1	Dense layer 10
	Flatten layer	LSTM layer 15	
	Dense layer 10	Dense layer 10	
Total parameters	97114	11440	8830

Table 3.1: Keras network model settings used for the three selected non-spiking implementations. The size and type of each layer are depicted, as are the total parameters of each design. The softmax activation function is used for each final dense layer. *For the Conv2D layers in the CNN implementation, a kernels size of 3x3 combined with the relu activation function is used.

4 Training methodology

In this chapter, a description of the training methods applied to the proposed architecture is given. The parameters, process flow and complications of the methods are highlighted. Furthermore, the underlying philosophy behind the chosen speech data is discussed.

4.1 Temporal spike training

The use of a temporal-based SNN has strong consequences for the applicable training methodologies. As discussed in Section 2.5, it prohibits the utilization of ANN to SNN conversion. A technique frequently used to create deep SNN architecture capable of performing complex tasks. Due to the time constraints, a feasible option is that cloud be implemented in time is the Tempotron [24] learning rule. Other more capable training methods are significantly more complex to implement, which could jeopardize the timeline. Tempotron is a supervised synaptic learning rule that enable neurons to efficiently discriminate different spatio-temporal spike patterns. Powerful, when applied to temporal spike trains where information is encoded in the exact timing of the spikes [24][39]. Like other learning rules that operate directly on the spike trains, the training capabilities are limited to single network layers. The Tempotron learning rule incorporates the leaky integrate-and-fire (LIF) model, for which the total membrane potential is described by Equation 4.1:

$$V(t) = \sum_i w_i A_i(t) + V_{rest} \tag{4.1}$$

where w_i described the weight of the i-th synapse and V_{rest} the resting potential of the neuron. The neurons total membrane potential is the weighted summation of the postsynaptic potentials (PSP) from all synapses. The accumulated PSP of the i-th synapses is described by Equation 4.2:

$$A_i(t) = \sum_{t_i<t} K(t - t_i)) \tag{4.2}$$

where K is the normalized PSP kernel when an input spike is received at time t_i. It describes the postsynaptic potential elicited by the incoming spike with parameters τ and τ_s denoting decay time constants of the membrane integration and synaptic currents, see Equation 4.3. The factor V_0 is used for the normalization of the postsynaptic potential kernel. When the potential exceed the threshold V_{th}, the neuron will enter its resting state by shunting all incoming spikes.

$$K(t - t_i) = \begin{cases} V_0 \left[exp\left(-\left(t - t_i/\tau\right) - exp\left(-\left(t - t_i/\tau_s\right)\right)\right)\right] & t \geq t_i \\ 0 & t < t_i \end{cases} \tag{4.3}$$

The binary classification of the input patterns is performed with the following outcome; a pattern which should elicit at least one postsynaptic action potential, or a pattern which should have no response accordingly. Initially, the neuron has no knowledge of which spike pattern belongs to which classification and has to learn it iteratively. This learning process is performed by adapting the synaptic weights w_i. In the first scenario, a spike pattern is presented which should lead to at least a postsynaptic potential, but the postsynaptic neuron did not fire, in this case all synaptic efficacies are increased by delta w_i. The next scenario, where a pattern should lead to no response but is followed with a postsynaptic response, in this case leads to a decrease of the synaptic efficacies by Δw_i. Δw_i can be described by Equation 4.4 where t_{max} is the time instance at which the postsynaptic potential $V(t)$ obtained its peak value, and λ specifies the learning rate.

$$\Delta w_i = \lambda \sum_{t_i < t_{max}} K\left(t_{max} - t_i\right) \tag{4.4}$$

4.1.1 Process flow

The Tempotron learning rule is implemented external to the simulation toolbox, reducing development complexity and time. As a result, extra operation are required to insert the trained model into the simulator. Each of the required steps to complete the training process flow are described in this section.

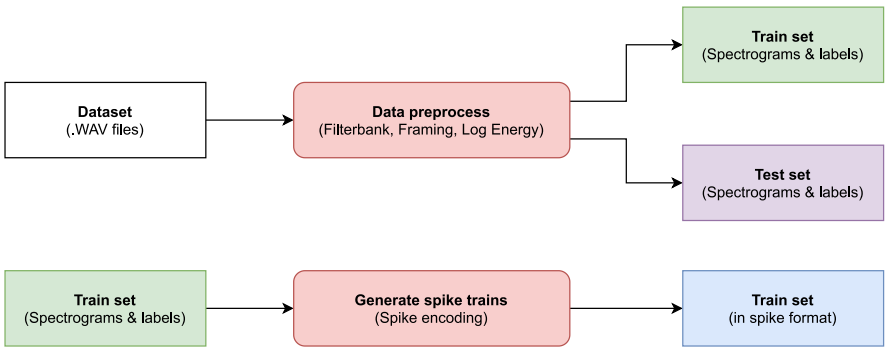


Figure 4.1: Step 1: Data preprocessing of the speech data, resulting in a set containing each spectrogram.

In first step, the speech data is preprocessed to minimize the simulation time of the model. Here the auditory front-end, where the audio is filtered and framed, results in set of spectrograms. See Section 3.1.1 for a more in depth description. The data is split into two parts, a train- and test set where the latter contains 20 percent of the entire dataset. Before the Tempotron learning rule can be applied, the training data must first be encoded into temporal spike trains. To ensure proper operation, the encoding settings used for Tempotron training need to be identical to the ones used in the simulator. After this additional step, the training data is in the spike format required for the learning stage.

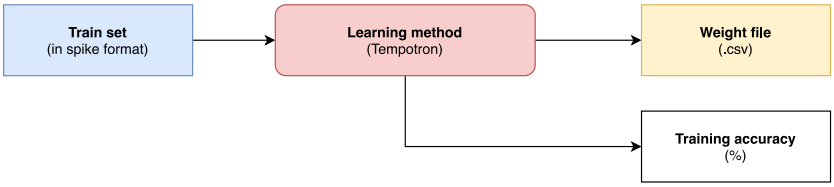


Figure 4.2: Step 2: Training the network model with help of the Tempotron learning rules.

In the second step, the training data is fed to the Tempotron learning method. Based on the included label the learning methods is capable of training the SNN model. The user specifies the learning-rate, epoch and dimension of the network layer. The training stage results in a weight file which contains the syntactic weights of each connection of the network. Furthermore, the acquired training accuracy is shown which can give some indication of performance.

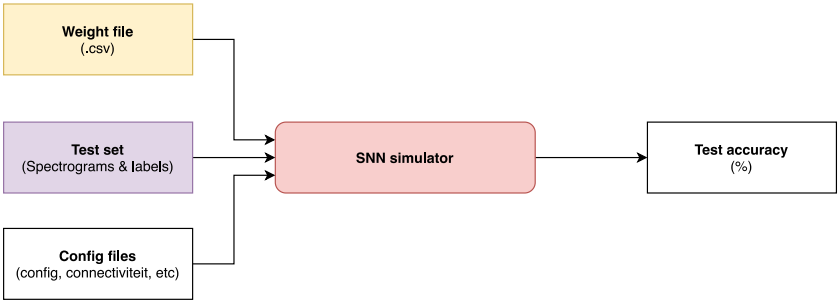


Figure 4.3: Step 3: Testing network performance with help of the simulation toolbox.

In the third and final step, the weight file is inserted into the simulation toolbox. Accompanied by network config files and the preprocessed training data. The configuration files include the neuron parameters, network partitions and dimension. Based in the provided data, the behavior of the SNN is simulated and interpreted by the decoder. This results in a performance figure of the architecture, including classification accuracy, latency of inference and number of spike processed by the network. The performance metrics described are used to evaluate the network's performance. The analyses of these metrics can be found in Chapter 5.

4.1.2 Complications

It is worth noting that the stand-alone application of the Tempotron learning rules presents a minor complication. The Tempotron rules is defined following continuous operation of the neurons and spike signals. These characteristics are thus also present in implemented training framework. In comparison, the simulation toolbox utilizes a discrete approach for computation efficacy. These two different approaches can therefore have negative effects on the obtained simulation performance. For example, the exact temporal placement of the spike could slightly differ between the two and thus

impact the performance. However, these problems will not occur when applied to the neuromorphic hardware itself since it operates in the continuous domain.

The ultimate objective is to incorporate the Tempotron learning rules fully into the simulation toolbox. By merging the implementations the opportunity arises to align both approaches, thus eliminating the described complication. Furthermore, it will simplify the workflow as the manual steps described in Section 4.1.1 can be integrated.

4.2 Speech data

Following the given requirements, an appropriate dataset needed to be selected which would mimic real-world conditions. The speech recordings thus include both clean and noisy data-samples. In order to verify the noise robustness of the proposed architecture, control over the amount of noise is beneficial. This enables the effects of certain amounts of noise to be thoroughly investigated.

4.2.1 Dataset candidates

Due to the infancy stages of the project, the philosophy to start small and slowly expand the complexity of the system is followed. This philosophy is reflected speech data used within the project, as dataset with a relative low complexity are explored. This type of audio data will provide a good starting point from which useful experience and insights can be extracted. The main dataset candidates are discussed in more detail below.

4.2.1.1 TIDIGITS

This large dataset has been collected to designing and evaluating algorithms for speaker-independent recognition of spoken digits ('0'-'9','oh'). It contains a large amount of speakers (326) spread over a wide variety of categories like age and gender. The vast majority of the dataset consists of digit sequences of varying lengths, but more importantly it also contains more than 7,000 isolated utterances each collected in a quit environment and digitized at 20 kHz. This fraction of the dataset of mainly used in the current state-of-the-art, e.g. [32][33][34]. This is beneficial since direct performance comparison can be made. The main drawback of dataset is its availability, since licensing and costs are involved. Furthermore, the dataset only contains clean speech samples which don't mimic real-world conditions.

4.2.1.2 Free Spoken Digit Dataset (FSDD)

Like the TIDIGTS dataset structure, the Free Spoken Digit Dataset (FSDD) [40] also consist of recorded spoken digits ('0'-'9'). FSDD is an open dataset, which means it will continue to grow over time as data is contributed. At time of this research, the dataset contains 2,000 recording spread over 4 speakers, thus per speaker 50 recordings of each digit. All samples are digitized at 8 kHz, recorded in ideal conditions where minimal background noise is present. The recordings contain only the spoken digits with the silence parts prior to and after trimmed. This dataset was selected because

it mimics the structure of the TIDIGIT dataset but is freely available, which allowed prompt progress of the project. Like TIDIGITS, it only contains speech samples with minimal background noise. This is solved by adding real-world background noise to the clean speech recordings, this process is described in more detail in Section 4.3.

The restricted size of the dataset shouldn't impose a problem since the long-term goal is to use a more use-case specific dataset, i.e. actual speech commands. Due to the infancy of this project, the aim of this project is to lay the foundation, thus the research of more use-case specific datasets is beyond the scope of this project.

4.2.1.3 Speech Commands

Taking the long-term objectives into account, some research has been carried out into datasets with higher complexity. A strong candidate is the Speech Commands dataset [41] which is composed of more meaningful speech recordings (35 words) including spoken digits ('0'-'9'). This large dataset contains 64,727 utterances from 1,881 speakers digitized at 16 kHz. The recordings reflect real-world condition including background noise, poor quality recording equipment and people talking in a natural, chatty way. This aspect can sadly not be configured, making it difficult to research the robustness under different noise levels. Given that this is an integral part of this research, it was decided not to use this dataset.

4.3 Testing for robustness

The selected dataset is recording under ideal condition with limited background noise. In order to research the effects of background noise, different levels of noise are added to the clean speech samples. Due to the specifics of the use-case, an audio clip which contains actual real-world background noise is added to the clean speech signals. In this case, restaurant background noise is selected as it covers similar noise characteristics. The original (clean) signal samples are altered with noise of varying signal-to-noise ratios (SNR) and are used to evaluate the performance of the proposed architecture. This method gives great control and flexibly over the amount of background noise that is exposed to the system so its effect can be thoroughly analyzed. The signal-to-noise ratio specifies the power ratio between the acoustic signal and ambient noise:

$$SNR_{db} = 10log\left(\frac{P_{signal}}{P_{noise}}\right) = 10log\left(\frac{RMS^2_{signal}}{RMS^2_{noise}}\right) \tag{4.5}$$

where P is the average power. Since the speech samples are discrete signals, the average power of both signals can be calculated using the alternative root means square (RMS) formulation. Based on the desired SNR value, a required RMS value for the noise signal can be found.

The noise audio clip is then modified so its RMS equals the previously obtained $RMS_{required}$. This is achieved by multiplying the whole signal (element-wise) by a constant a, which can be found using the following equation:

$$\sqrt{\sum \frac{(a * n_i)^2}{n}} = a * RMS_{noise} = RMS_{required} \tag{4.6}$$

The modified noise audio clip can now be added to the clean speech signal, resulting in an audio clip that contains the desired SNR amount of background noise. To prevent that each audio clip contains the same background noise, a random section of background noise is selected. This is possible since the restaurant background noise recording has a duration of more than 45 seconds. This process prevents that the network to learns the exact noise signal which would result in unrealistic network performance. The flow diagram of the whole process is depicted in Figure 4.4.

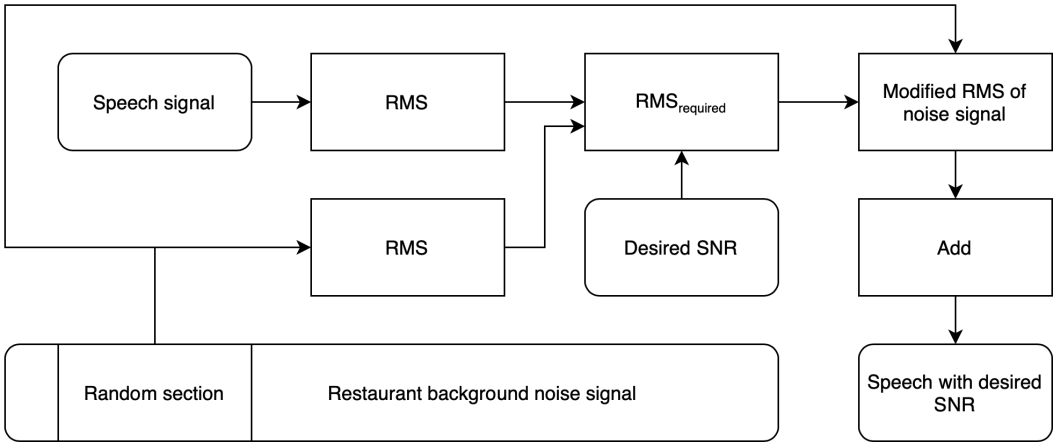


Figure 4.4: Process flow diagram where a random section of background noise is added to a clean speech signal, based on desired SNR value.

5 Results

In this chapter the performed experiments and the results obtained are discussed in detail. The performance of the architecture and training methods applied are analyzed. The exact neuron parameters used for all simulation can be found in A.2, which are based on intended neuromorphic hardware platform. In the first part, the overall classification performance of the network is analyzed under ideal condition, thus including hardly any background noise. Different network sizes are explored and the performance impact is discussed. In the second part, the impact of background noise is studied, here specific levels of SNR background noise are used. At last, a comparison with traditional non-spiking neural networks is conducted. Important performance metrics like classification performance, power requirements and time to inference are analyzed.

5.1 Impact of network size

To find the optimal architecture, in terms of the balance between performance and size, network architectures of different sizes are examined. Since the network size is fully dependents on the encoder used, this part is split in two parts. In the first part population threshold encoding is utilized, where a variety of population sizes (threshold values) are analyzed. In the second part, the enhanced SOM encoded architecture is examined. The network size depends on the size of the SOM layer. All tests are conducted on the unmodified clean dataset [40], which represent the ideal conditions.

5.1.1 Population Threshold encoding

In this first implementation, a population threshold encoder is used to encode each frame of the spectrogram into a spike pattern. The critical parameters is the size of the population, and thus the number of threshold values used. The linearly scaled threshold values are used to encode the incoming information stream, each values resulting its own spike train. For proper functioning, a sufficient amount of threshold values is required to capture features embedded in the input data. In order to find the optimal value, a broad range of thresholds sizes is explored. In Table 5.1 the exact threshold sizes used for testing are shown, including the required number of neurons for the input layer of the Spiking Neural Network. Each spectrogram frame fed into the encoder contains 20 frequency bins and is encoded by both the onset and offset neurons. This leads to the specified required amount of neurons needed for the input layer of the Spiking Neural Network. The specific test sizes are used to verify the performance of the subsequent SNN in terms of classified accuracy. This performance metric is used to determine the optimal number of thresholds used to encode the incoming acoustic data.

#Encoder thresholds	#Onset neurons	#Offset neurons	#Neurons in input layer
3	60	60	120
6	120	120	240
9	180	180	360
12	240	240	480
15	300	300	600
18	360	360	720
21	420	420	840

Table 5.1: Overview of the range of thresholds applied, including the total neurons required for the input layer of the SNN classifier, which are split into onset and offset neurons 3.1.2. The numbers are based on the spectrogram frame size, which are composed of 20 frequency bins.

In Figure 5.1 the performance impact over a wide range of threshold size is depicted. The effects of an insufficient amount of threshold is eminent, as the performance suffers greatly with deficient population sizes. For example, the encoder with only 3 thresholds (120 input neurons) is unable to capture all important features embedded in the variations of the input signal, and thus the classification performance is deteriorating. In addition, significant performance gains made by simply doubling the population size. This finding confirms the need for sufficient thresholds, due to the major impact on performance.

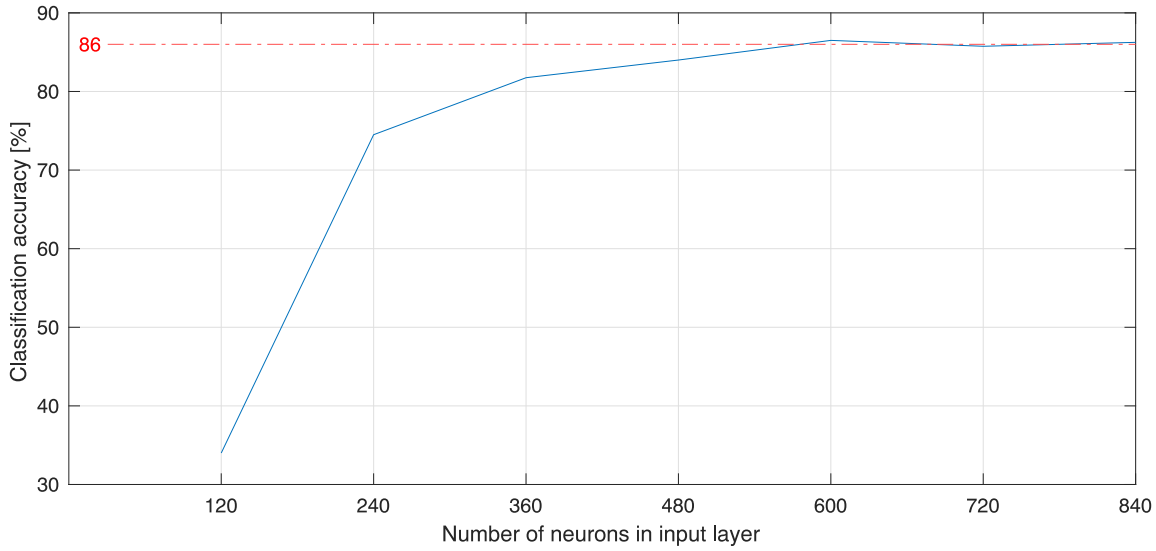


Figure 5.1: Overview of the performance impact, in terms of classification accuracy, when different amounts of thresholds are applied to encode the input data. The number of threshold and the corresponding neuron count can be found in Table 5.1. The exact results can be found in Appendix B.1.1.

Furthermore, the results shows that no performance gains are made after 15 thresholds (600 input neurons) as the classification accuracy converges to around 86%. The performance under ideal conditions (limited background noise) no longer scales with the population size. Continuing to increase the sensitivity of the encoder does not provide new temporal information that is of use for the SNN classifier. That is under the ideal conditions, it could potentially help improve the performance under noisy conditions as is explored in 5.2. The performance convergence may be due to the SNN's inability to filter out lesser important features, as they still negatively effect the outcome. Deeper SNN architecture cloud improve the filter capabilities of the network, as the architecture is currently limited two layers (a fully connected input and output layer).

The tests carried out in the Section 5.1.2 aims to verify the need for a filtering stage, allowing the subsequent SNN classifier to focus on the more important features, as feature reduction can improve performance.

5.1.2 SOM layer encoding

As previously discussed in 5.1.1, the classification accuracy converges as increasing the sensitivity of the threshold encoder is only effective up to a certain point. This is due to properties of the encoder utilized, as it tries to capture all features. In case of high sensitivity, the balance between important and less important features shifts as more minor features are captured. This shift negatively impacts the performance of the subsequent SNN classifier. The limited depth of the SNN classifier lowers its filtering capabilities and thus the performance gains halting. To support these claims an enhanced architecture is researched, which employs a SOM for feature encoding. The SOM provides filtering capabilities useful for feature reduction. The reduced space facilitates the task of the SNN classifier, as it decreases the load of the classifier.

SOM dimensions	#Neuron in input layer	SOM dimensions	#Neuron in input layer
2x2	4	14x14	196
4x4	16	16x16	256
6x6	36	18x18	324
8x8	64	20x20	400
10x10	100	22x22	484
12x12	144	24x24	576

Table 5.2: An overview of the exact SOM dimension tested with the enhanced network architecture, previously described in 3.2.1. These dimensions are used to verify the improved network performance against the initial threshold encoding architecture.

A wide variety of dimension for the SOM layer is explored and its performance impact is studied in order to find the optimal network configuration. In terms of classification accuracy and network size. To limit the research scope, only the quadratic dimensions of the SOM are explored. The exact dimension used for testing the classification performance can be found in Table 5.2. All test results shown in Figure 5.2 are obtained under ideal conditions where little no background noise is present in the

input data. The results shown that significant performance gains are made when a sufficiently sized SOM layer is utilized. As with previous architectures, the classification accuracy converges, but in this case at a much higher boundary of 97%. This boundary is reached starting from 324 neurons (18x18 SOM) as marginal gains are obtained by larger sized SOM implementations. As a side note, the tests are conducted under ideal conditions, the larger sized SOM implementations could potentially help improve the performance under noisy conditions as is explored in Section 5.2.

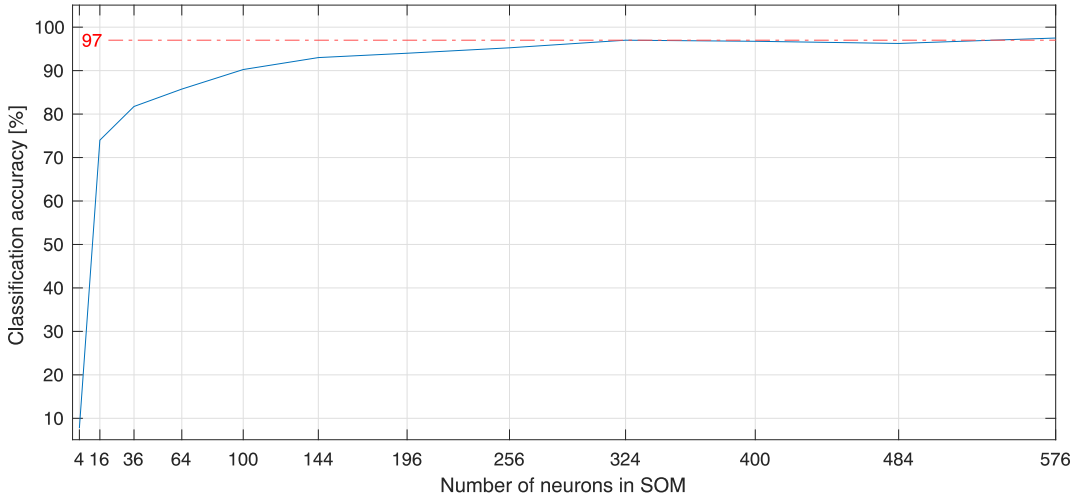


Figure 5.2: The performance impact, in terms of classification accuracy, for a variety of SOM dimensions used to encode in the input data. An overview of the exact dimensions and the required amount of input neurons can be found in Table 5.2. The exact results can be found in Appendix B.1.2.

The performance gains made shows that the SNN classifier benefits from a reduced feature space, allowing it to focus on the more important features. A gain of 11% is achieved when compared to the initial threshold encoded architecture. It highlights the effectiveness of the SOM layer to form low-level feature representations. Furthermore, the required number of neurons in the input layer of the SNN is greatly reduced. The threshold encoded architecture requires at minimum 600 input neurons to reach it performance boundary of 86%. In contrast, the SOM encoded architecture requires only 324 input neurons to reach its boundary at 97%. However, the input size reduction of the SNN classifier comes at a computational cost. This due to the utilization of a non-spiking SOM, where even for inference a large amount of computations is required to find the BMU. The added computational load negatively impacts the power requirements of the whole architecture, as is show in Section 5.5. The use of a non-spiking SOM is only studied to see if the network would benefit from this additional layer. Other studies [42] [43] have demonstrated the viability of a Spiking SOM, which mimics the competing behavior of an ordinary non-spiking SOM. This solution allows for a fully spiking architecture where the SOM layer is retained, but its computational load is omitted. This promising direction isn't fully explored due to time constraints, a more detailed description is given in Chapter 7.

5.2 Impact of noise

In the previous section, the classification accuracies of the proposed network architectures is analyzed, focussing only on clean speech samples. These samples represent the best case scenario where a minimum amount of background noise is present. An important requirement for a robust ASR system is its performance in more realist conditions, where different level of background may be present. In this section, the robustness to background noise of each architecture is studied. A range of SNR level of background noise are added to the input signals and the performance impact is analyzed, see Table 5.3 for the specific values.

SNR [dB]	Description
20	Low, slight addition of noise
10	Medium, more speech than noise present
0	High, equal amount of speech and noise present
-5	Severe, more noise than speech present

Table 5.3: The specific SNR levels applied to the clean speech samples. Used for testing robustness of the proposed architectures.

The impact of each SNR level of background noise is tested separately, and thus gives an indication of performance in different noisy conditions. The results are obtained with help of two types of training strategies, namely:

- **Mismatched conditions (MM)**, where the network is trained with clean speech samples only representing ideal condition where minimal background noise is present. In contrast, the data used for testing does contain samples with selected levels of background noise.
- **Matched conditions (MC)**, both the training and testing data contain noisy speech samples. This method better prepares the network to handle background noise and helps counteract the negative impact on performance.

5.2.1 Population Threshold encoding

In this section the robustness to noise is studied for the threshold based architectures. The three top performing threshold based architecture are selected, each reaching the performance boundary under ideal conditions (clean of background noise). Specifically, the included architectures require 600 (15 thresholds), 720 (18 thresholds) and 840 (thresholds) input neurons. As shown in 5.1.1, marginal performance gains are made after utilizing more than 15 threshold values. The two larger designs are included to discover the potentially beneficial impact under noisy conditions. In Figure 5.3 the performance under different SNR levels is depicted. In addition, the previously obtained results under the ideal condition are included. The results shown the impact of background noise on the overall classification accuracy of the network.

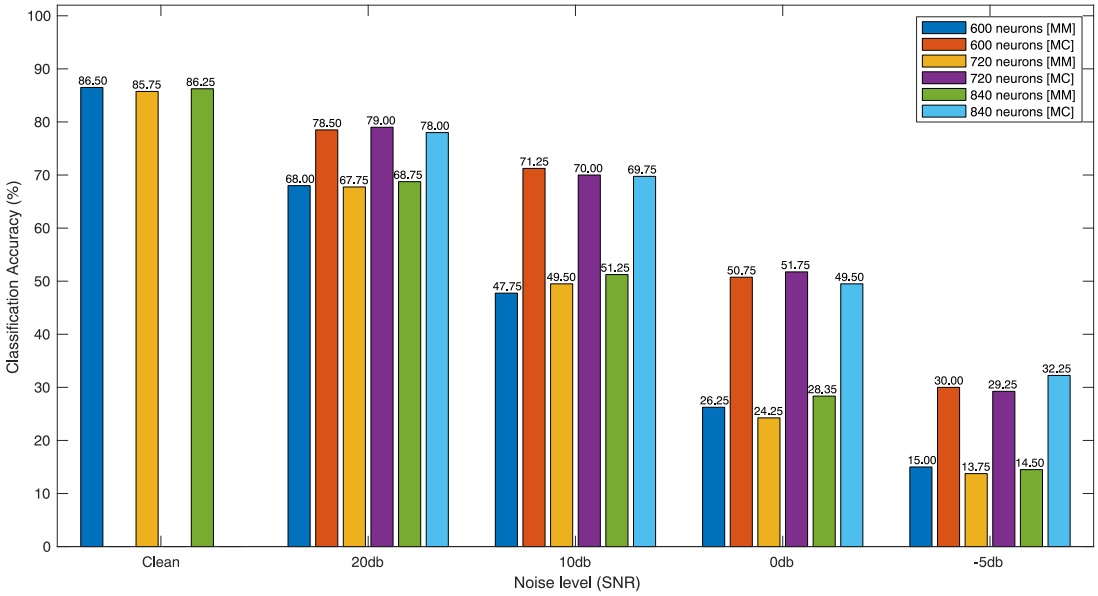


Figure 5.3: The performance results obtained for the architecture utilizing population threshold encoding. Specific SNR levels are tested and aim to mimicking real-world environments. It illustrates the performance impact for variety of noise levels, where lower dB values correspond the harsher conditions. The exact results can be found in Appendix B.2.1.

The performance gains made when training the network with matched conditions (MC) is immediately apparent, in some cases doubling the performance for missmatched conditions (MM). The performance impact already notable when only a low amount of background noise (20 dB) is present, showing a deterioration of 10%. At higher levels of background noise the deterioration becomes more eminent. This outcome is related to specific properties of the threshold encoder, as it requires a sufficient amount of thresholds to properly capture the features embedded in the input signal. Here, a high amount of thresholds leads to a heightened degree sensitivity. This property negatively impacts the robustness of the architecture, as the resulting spike trains are susceptible to alterations due to noise. In Table 5.4 the alterations of the spike trains are highlighted by the average spike count per sample.

Architecture	Average spike count per sample				
	Clean	20 dB	10 dB	0 dB	-5 dB
600	464	487	529	628	668
720	563	592	644	761	810
840	663	697	756	896	950

Table 5.4: An overview of the impact of background noise on the threshold encoded spike trains. The average spike count highlights the alterations due to the noise sensitivity of the threshold encoder. Due to the limited filtering capabilities of the low depth SNN, this increased amount of less important spikes deteriorates the network performance.

The balance between important and less important features is shifted, increasing the workload of the SNN classifier. This combined with the limited filtering capabilities of the SNN, due to the low depth of neural network, hinders its performance in more difficult conditions. Furthermore, like the results obtained under ideal conditions, little to no performance gains are made by the larger sized networks. Based on results, the architecture with 15 thresholds (600 input neurons) has proven to be optimal in terms of network size and classification performance. This since the performance boundary is met in both clean and noisy conditions and the classification accuracy is equivalent to much larger sized network iterations.

5.2.2 SOM layer encoding

The same procedures, as previously discussed in 5.2.1, are applied to the enhanced network architectures which utilize a SOM for encoding. The four top performing architecture are selected, each reaching the performance boundary under ideal conditions (clean of background noise). Specifically, the included architectures require 324 (18x18 SOM), 400 (20x20 SOM), 484 (22x22 SOM) and 576 (24x24 SOM) input neurons. As shown in 5.1.2, marginal performance gains are made when increasing the dimension of the 18x18 SOM. However, these larger designs are included to discover the potentially beneficial impact under noisy conditions. In Figure 5.4 the performance results under different SNR levels is depicted. In addition, the previously obtained results under ideal condition are included.

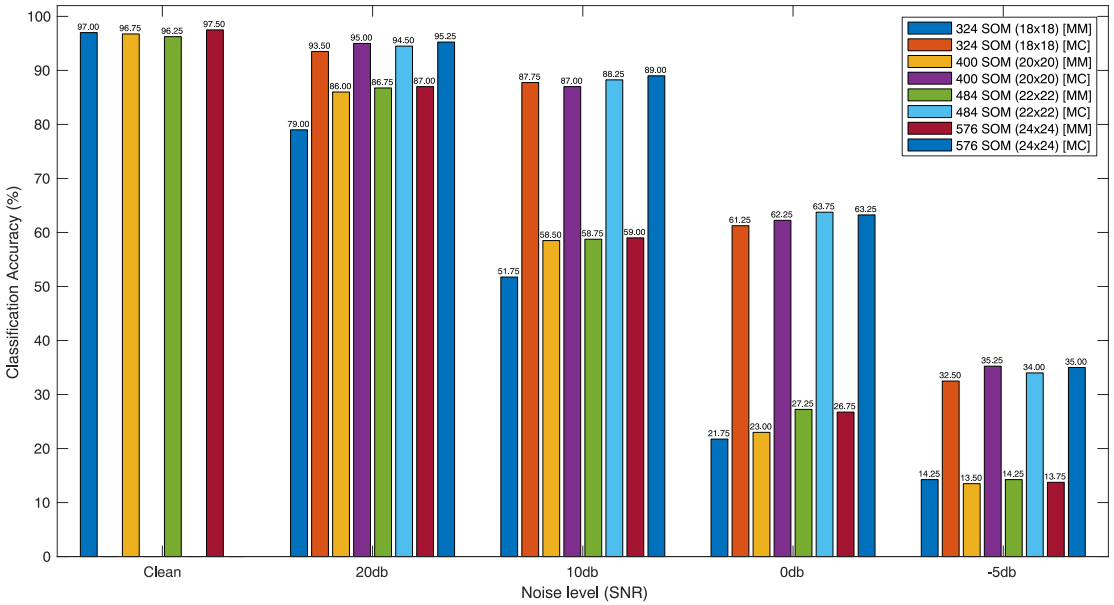


Figure 5.4: The performance results obtained for the SOM enhanced architecture. Specific SNR levels are tested and aim to mimicking real-world environments. It illustrates the performance impact for variety of noise levels, where lower dB values correspond the harsher conditions. The exact results can be found in Appendix B.2.2.

The results once again emphasize the performance gains achieved when the network is trained based on the matched conditions. Like under clean conditions, enhanced performance figures are obtained under noisy conditions. For example, the performance under 20 dB of noise is greatly improved with these network architectures. In this case only a performance drop of 4% is observed, compared to the 10% drop observed for the threshold encoded network design. The improved classification accuracy is assisted by the SOM, as the spike trains are less severely altered by the background noise. As shown in Table 5.5, the average spike count is consistent under each noise level. These properties reduce the (filtering) load on the SNN classifier, thus improving the overall performance.

Architecture	Average spike count per sample				
	Clean	20 dB	10 dB	0 dB	-5 dB
324 SOM (18x18)	423	424	431	439	446
400 SOM (20x20)	421	423	429	440	447
484 SOM (22x22)	419	422	429	440	448
576 SOM (24x24)	418	422	428	440	447

Table 5.5: An overview of the impact of background noise on the SOM encoded spike trains. The more constant average spike count indicates less drastically altered spike trains produced by the SOM. This steady behavior will reduce the workload of the subsequent SNN classifier, as can be seen by the improved classification accuracy in Figure 5.4.

It is important to note that no gains are made under the severely noisy conditions of -5 dB. The classification accuracy obtain under these conditions are in the same range as the threshold encoded architecture. Based on human observation, the conditions of -5 dB are described as hard to classify. The low performance under this condition can therefore be attributed to this high level of difficulty. The use of an array of microphone in these conditions will significantly help the performance under these conditions. It allows the utilization of noise-cancelling techniques which improve the robustness of the implementation. This solution is of interested in future works as they are beyond the scope of this research project.

5.3 Non-spiking neural network performance

All results obtained by them self have minimal meaning without directly comparing them with more traditional ANN architectures. The more mature and widely used ANN architectures offer a perspective in terms of achievable performance. The included commonly used network architectures are the Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and the Long Short-Term Memory (LSTM). The exact parameters and network layers used are discussed in 3.3. All networks are trained on exactly the same data, meaning the spectrogram of each audio sample. The classification accuracies obtained for each of the network architectures are shown in Figure 5.5.

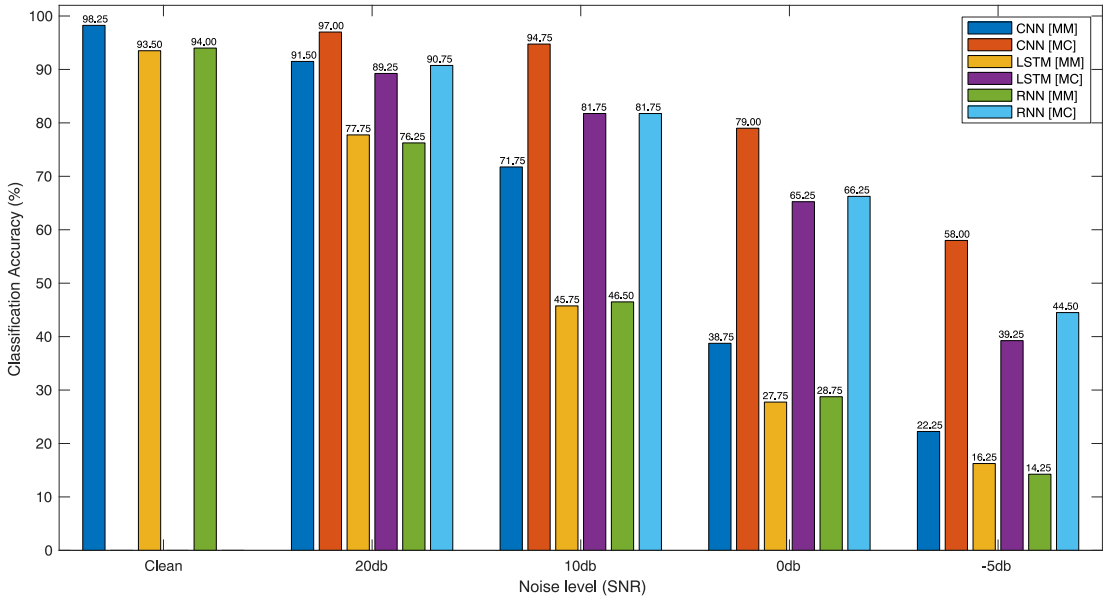


Figure 5.5: An overview of the classification performance for traditional ANN architectures. These results are obtained in a variety of conditions including; clean and SNR levels of added background noise.

Similar to the spiking domain, the benefits of training the networks with help of matched conditions (MC) is eminent. As the SNR level decreases, the effectiveness of this method increases as significant gains are achieved in these conditions. The utilization of matched conditions is thus highly effective for both spiking and non-spiking neural networks. Just like the spiking implementations, the classification accuracy starts to decreasing from 10 dB onwards. This effect applies to a lesser extent to the CNN architecture, since its performance is less hindered in these conditions. The performance degradation becomes more evident from 0 dB onwards. This is especially the case for both the LSTM and RNN networks. As demonstrated by the CNN, the temporal information is less critical compared to spatial information. This is mostly due to the set limitations of the dataset applied. As for this project, the initial focus is set on single word classification. This leads to a reduced amount of temporal information, the more static properties of the dataset allows the CNN to shine. The

performance difference between the LSTM and RNN are in most cases negligible, but in the extreme condition of -5 dB the RNN outperforms the LSTM. In addition, the performance differences can be related to the network size. This since the CNN is significantly larger in terms of trainable parameters, in some case by a factor of 10 (see Table 3.1). Larger sized LSTM and RNN architectures are tested, but did not result in groundbreaking performance leaps. Thus, in terms of classification accuracy, the CNN architecture achieve the best performance under tested all conditions.

In Table 5.6, the obtained classification results for both spiking and non-spiking architecture is given. It includes only the results obtained by applying the matched-condition training method, as it achieve a higher classification performance. For the spiking architectures, the network designs with minimum dimensions that reaches the top performance boundary are selected. Specifically, the threshold architecture (PTA-SNN) with a total population size of 600 (15 threshold values) and the SOM 18x18 enhance architecture (SOM-SNN). When comparing all network architectures, it can be seen that in the clean and less harsh conditions of 20 dB and 10 dB SNR, the SOM-SNN produces results comparable to the top performing CNN architecture. The SOM encoder with the subsequent SNN classifier (SOM-SNN) outperforms the LSTM, RNN and PTA-SNN implementations. The low-level feature extraction significantly boost performance, as in all cases it outperforms the PTA-SNN implementation. In the more harsh conditions, the CNN is still outperforming all other network designs. In these conditions, better classification accuracies are obtained by larger sized SOM-SNN architecture. For example, the SOM 20x20 implementation achieve and accuracies of 35.25% in the most extreme conditions of -5 dB SNR, bringing its performance in range of the LSTM architecture. All things considered, the results show that the spiking implementation can reach CNN like performance in terms of classification accuracy, especially in more forgiving conditions. This while its network size is relatively small compared to the CNN architecture.

SNR	CNN	LSTM	RNN	PTA-SNN	SOM-SNN
Clean	98.25%	93.50%	94.00%	86.50%	97.00%
20 dB	97.00%	89.25%	90.75%	78.50%	93.50%
10 dB	94.75%	81.75%	81.75%	71.25%	87.75%
0 dB	79.00%	65.25%	66.25%	50.75%	61.25%
-5 dB	58.00%	39.25%	44.50%	30.00%	32.50%

Table 5.6: The overall classification accuracy of the tested network models trained with matched condition training. For the spiking implementations the top performing threshold architecture (PTA-SNN) with a total population size of 600 (15 threshold values) and the SOM 18x18 enhance architecture (SOM-SNN) are selected.

The results in Table 5.6 shows that all architectures perform poor under the extreme conditions of -5 dB. This could be the results of the method to used add background noise to the audio samples, described in 4.3. A random section of the background noise recording is selected, this signal is not stationary and may contain different noise levels. The average SNR is calculated of the selected section, which could affect the

actual SNR level. These effects will be most notable under harsh conditions and may negatively impact performance. More precise tests are performed by adding white Gaussian noise to the audio samples, where the noise has uniform power across the frequency bands. The results obtained are shown in Table 5.7. As suspected, the performance indeed improves under the more harsh conditions. The outcome however is still identical as the CNN implementation still outperforms all other implementations. The SOM-SNN is still in proximity of the CNN performance under clean and less noisy conditions. In all other conditions is performance is in the same proximity is the LSTM and RNN architectures. It is suspected that the relatively simple decoders hurt the performance of both SNN architectures. Due to the shallow SNN utilized, noise can easily alter the output with only a single added spike. Thus changing the first arrival of a spike, or output frequency which will lead to misclassification. More capable decoders, possibly small ANN classifiers could significantly improve performance under noisy condition. However, this increase in performance will have a negative impact on the power requirements of each solution.

SNR	CNN	LSTM	RNN	PTA-SNN	SOM-SNN
Clean	98.25%	93.50%	94.00%	86.50%	97.00%
20 dB	95.35%	89.25%	89.75%	81.50%	92.75%
10 dB	94.00%	82.25%	83.50%	75.25%	84.35%
0 dB	83.00%	74.50%	74.50%	53.00%	65.25%
-5 dB	71.25%	58.75%	58.25%	36.75%	54.75%

Table 5.7: Results obtained with added white Gaussian noise. For the spiking implementations the top performing threshold architecture (PTA-SNN) with a total population size of 600 (15 threshold values) and the SOM 18x18 enhance architecture (SOM-SNN) are selected.

5.4 Distinguish between speech and noise

In this section, the ability of the network to distinguish speech and background noise is studied. This important characteristic is tested by feeding only background noise to the network. For this analysis the two top performing network architectures are selected, this in terms of classification accuracy and effective network size. To be more specific, these are the threshold architecture (PTA-SNN) with a total population size of 600 (15 threshold values) and the SOM 18x18 enhance architecture (SOM-SNN). Both network architecture perform poor when only noise is presented to the network, and is unable to distinguish speech from background noise. This behaviour is illustrated in Table 5.8, which contains the confusion matrix obtained for the SOM-SNN architecture. The confusion matrix for the PTA-SNN architecture, and for its enhanced state can be found in Appendix B.3.

		Predicted										
		0	1	2	3	4	5	6	7	8	9	**X**
Actual	0	1	0	1	0	37	1	0	4	0	0	**0**
	1	4	2	1	0	37	0	0	3	0	0	**0**
	2	3	3	1	0	31	3	0	6	0	0	**0**
	3	3	0	1	0	36	2	0	2	0	0	**0**
	4	7	0	0	0	33	3	0	2	0	0	**0**
	5	2	2	0	0	34	3	0	3	0	0	**0**
	6	5	1	0	0	31	4	0	2	0	0	**0**
	7	4	0	2	0	34	3	0	3	0	0	**0**
	8	7	1	1	0	33	1	0	1	0	0	**0**
	9	7	1	2	0	30	4	0	2	0	0	**0**

Table 5.8: Confusion matrix for the SOM-SNN architecture when only background noise is presented to the network. Since the network is trained to classify the speech samples, no response from the system is desired as any output activity corresponds to a speech class. However, this is not the case, as all samples results in misclassification of speech instead of noise.

As the network is trained to only classify selected speech features, no spike activity at the network output is desired. However, this is not the case, as a large amount of output spikes results in misclassification. It is suspected that the frequency components in the noise samples are highly active in a certain bin, which the subsequent network recognizes as a key feature of class 4. This is caused by the network design where each output is associated with a specific speech class. This obstacle is overcome by adding a dedicated noise class to the SNN classifier, as is discussed in 3.2.2. This relatively simple addition to the network architecture significantly improves performance, as shown in Table 5.9. In both cases, the enhanced network architectures (added noise class) are highly capable of distinguishing between speech and noise. See Table 5.8, which contains the improved confusion matrix obtained for the SOM-SNN architecture.

Furthermore, the performance penalty of the added class is minimal as the classification accuracy under clean conditions is nearly identical. For both architectures, a decrease in performance of 0.5 percent or less is noted. The insignificant impact on the clean network performance is encouraging, but doesn't represent the system as a whole. In order to get the full picture, its impact under noisy conditions is as critical. The same approach used in 5.2 is applied, where different SNR level of background noise are presented to the network. An overview of the performance impact is depicted in Table 5.11. In order to make comparison more manageable, only the results obtained with matched-conditions training are included.

Conditions	Original		Added noise class	
	PTA-SNN	SOM-SNN	PTA-SNN	SOM-SNN
Clean	86.50%	97.00%	86.25%	95.50%
Noise detection	1.25%	0%	99.00%	100%

Table 5.9: An overview of the initial and enhanced network performance, when only noise is applied to the both architectures.

		Predicted										
		0	1	2	3	4	5	6	7	8	9	10	X
Actual	0/9	0	0	0	0	0	0	0	0	0	0	0	0
	10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**400**	**0**

Table 5.10: Confusion matrix for the SOM-SNN architecture when only background noise is presented to the network. The added class 10 corresponds to the noise, and it highly detected. The overall classification accuracies obtained can be found in 5.9. The confusion matrix when audio samples are presented can be found in Appendix B.3.5.

In general a negative performance impact is observed when the distinct noise class is added to the network. However, the affect are marginal in case of 20 dB and 10 dB SNR. In the more harsh conditions, especially in the 0 dB case, the effects are more pronounced. In some instances, the network classifies the noise corrupted speech samples as noise itself. However, this disadvantage is minor and the resulting benefits are of greater importance. It allows a continuous stream of audio data to be processed with little to no response when only noise is present, as shown by the results.

SNR	PTA-SNN	PTA-SNN*	Delta	SOM-SNN	SOM-SNN*	Delta
20 dB	78.50%	77.25%	-1.25%	93.50%	94.25%	-0.75%
10 dB	71.25%	70.00%	-1.25%	87.75%	86.75%	-1.00%
0 dB	50.75%	45.50%	-5.25%	61.25%	57.75%	-3.50%
-5 dB	30.00%	28.25%	-1.75%	32.50%	30.50%	-2.00%

Table 5.11: An comparison in classification performance between the initial architecture and the addition of a dedicated noise class. The difference in classification accuracy is given by the performance Delta.
*The same architecture but with the addition of a dedicated noise class.

5.5 Power consumption & latency

– Section removed due to confidentiality –

– Section removed due to confidentiality –

– Section removed due to confidentiality –

– Section removed due to confidentiality –

– Section removed due to confidentiality –

6 Conclusion

In this work, two promising SNN based architectures are proposed capable to detect and classify audio data. The architectures include an auditory front-end, feature representation learning and temporal classification. The key difference between the two architectures is the encoding method applied to convert the input data into temporal spike trains. The resulting temporal spike trains are fed to the subsequent single layer SNN classifier. In order to train the temporal SNN classifier effectively, an existing training method was selected and implemented. The supervised Tempotron learning rules are used to obtain the weights which are utilized by the SNN classifier. Furthermore, the training method has been successfully applied to other in-house research projects [44].

The incoming acoustic signal is preprocessed using a biologically inspired auditory front-end, performing low-level feature extraction. This process mimics the functionality of the human cochlea. A mel scaled filter is used to emulate the non-linearity of the human perception of sound, as it decomposes the incoming audio signal into different frequency bins. The signals in each frequency bin are divided into (overlapping) frames, after which the power spectrum of each frame is calculated. This process results in a spectrogram which is normalized to a set range. Before each spectrogram is fed to the subsequent SNN classifier, an encoder is needed to translate all embedded information into temporal spike trains. Two promising encoding options are explored, a commonly used population based threshold encoder and a self-organizing map (SOM). The latter improving feature separation by clustering data in an unsupervised fashion. Furthermore, it performs feature reduction as each spectrogram frame only results in a single output spike, thus achieving sparse spatio-temporal spike trains.

The potential of each SNN architecture is verified based on the classification accuracy, power consumption and latency for inference. For additional validity, the results obtained are benchmarked against commonly used ANN architectures. The results show that the SNN based architecture can closely match the top performing ANN models, especially under ideal conditions where a classification performance of 97% is achieved. This all while achieving significantly low power consumption. The utilization of a SOM has show to improve the overall classification performance, however due to its non-spiking characteristics it has a significant negative impact on the power consumption and latency. Other studies [42] [43] have demonstrated the feasibility of the spiking-SOM, which would overcome these obstacles.

In order to test for robustness, the impact of a wide variaty of background noise levels is studied. Here, audio samples recorded under ideal condition with minimal background noise are corrupted to a set SNR. In most cases, the SNN-based architectures can compete with the included ANN models. However, in extremely noisy conditions the shallow depth of the SNN classifier becomes more apparent as the classification performance suffers compared to deeper ANN architectures. It is suspected

that the relatively simple decoders hurt the performance SNN classifier. Combined with the shallow depth of the SNN classifier, noise can easily alter the output with only a single added spike. Thus changing the first arrival of a spike, or output frequency which will lead to misclassification. More capable decoders, possibly small ANN classifiers could significantly improve performance under noisy condition. However, this increase in performance will have a negative impact on the power requirements of each solution.

7 Future work

The obtained results demonstrate that the proposed architectures are highly capable of performing speech classification tasks. Nevertheless, potential improvements could be of interest and push the solution to the next level.

At this time, only the viability of a non-spiking SOM is studied. The power and latency estimations have also uncovered the negative properties of the non-spiking SOM. Other studies [42] [43] have demonstrated the viability of a Spiking SOM. When coupled with an appropriate encoding method, the behavior of traditional SOM can be mimicked. This eliminates the high computational load of the current implementation while retaining the favorable characteristics.

An important next step is to study or develop methods that are capable of training deep temporal SNNs. The current SNN classifier contains only an input and output layer, thus no hidden layer. The limited depth of the network could hamper its ability to perform more complex recognition tasks. It would potentially also improve the robustness of the solution, since in its current form, noise can easily alter the output of the network, reducing the classification accuracy.

In other important direction is the creation of a use-case specific dataset. This dataset should include useful speech recordings, for example spoken keywords that fit the intended application domain. By creating this data set, a setup with multiple microphones could be easily achieved. The use of a multiple microphone setup will open up new opportunities like noise cancellation, speaker localization, etc. These additional features make the solution more attractive to a broader market.

Subsequently, the realization of a hardware based auditory front-end is of great importance. This since the result are obtained with no hardware constraints. In the real world, restrictions do apply to the auditory front-end, which could affect its behaviour. A highly performing low power solution should be studied, and the functionality when combined with the subsequent SNN classifier needs to be verified.

Furthermore, it would be interesting to examine the classification performance when applied to a sequence of words. Due to the initial stage of this work, the focus has been on relatively simple classification tasks. The next step would be to increase the complexity of the classification task. A sequence of spoken words would be the ideal candidate, since the processing of temporal information should highlight the favorable properties of a Spiking Neural Network.

Bibliography

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, 1943, ISSN: 00074985. DOI: 10.1007/BF02478259.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986, ISSN: 00280836. DOI: 10.1038/323533a0.

[3] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A Stochastic Computational Multi-Layer Perceptron with Backward Propagation," *IEEE Trans. Comput.*, 2018, ISSN: 00189340. DOI: 10.1109/TC.2018.2817237.

[4] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, 2019, ISSN: 14764687. DOI: 10.1038/s41586-019-1677-2.

[5] S. Herculano-Houzel, *The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost*, 2012. DOI: 10.1073/pnas.1201895109.

[6] D. B. Tower, "Structural and functional organization of mammalian cerebral cortex: The correlation of neurone density with brain size. Cortical neurone density in the fin whale (Balaenoptera Physalus L.) with a note on the cortical neurone density in the Indian elephan," *J. Comp. Neurol.*, 1954, ISSN: 10969861. DOI: 10.1002/cne.901010103.

[7] F. A. Azevedo, L. R. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. Ferretti, R. E. Leite, W. J. Filho, R. Lent, and S. Herculano-Houzel, "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain," *J. Comp. Neurol.*, 2009, ISSN: 00219967. DOI: 10.1002/cne.21974.

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, 1952, ISSN: 14697793. DOI: 10.1113/jphysiol.1952.sp004764.

[9] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S. C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, *Neuromorphic silicon neuron circuits*, 2011. DOI: 10.3389/fnins.2011.00073.

[10] L. F. Abbott, *Lapicque's introduction of the integrate-and-fire model neuron (1907)*, 1999. DOI: 10.1016/S0361-9230(99)00161-6.

[11] L. de Gelder, "Population Step Forward Encoding," PhD thesis, 2021.

[12] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, no. AUGUST, pp. 1–9, 2015, ISSN: 16625188. DOI: 10.3389/fncom.2015.00099.

[13] R. B. Stein, E. R. Gossen, and K. E. Jones, *Neuronal variability: Noise or part of the signal?* 2005. DOI: 10.1038/nrn1668.

[14] N. K. Kasabov, *Time-Space, Spiking Neural Networks and Brain-Inspired Artifical Intelligence*. 2019, ISBN: 9783662577134.

[15] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *J. Comput. Syst. Sci.*, 2004, ISSN: 00220000. DOI: 10.1016/j.jcss.2004.04.001.

[16] Y. Dan and M. M. Poo, *Spike timing-dependent plasticity of neural circuits*, 2004. DOI: 10.1016/j.neuron.2004.09.007.

[17] R. Kempter, W. Gerstner, and J. L. van Hemmen, "Hebbian learning and spiking neurons," *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, 1999, ISSN: 1063651X. DOI: 10.1103/PhysRevE.59.4498.

[18] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nat. Neurosci.*, 2000, ISSN: 10976256. DOI: 10.1038/78829.

[19] Y. Dan and M. M. Poo, *Spike timing-dependent plasticity: From synapse to perception*, 2006. DOI: 10.1152/physrev.00030.2005.

[20] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, 1998, ISSN: 02706474. DOI: 10.1523/jneurosci.18-24-10464.1998.

[21] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biol. Cybern.*, 2008, ISSN: 03401200. DOI: 10.1007/s00422-008-0233-1.

[22] Y. Cao, Y. Chen, and D. Khosla, "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition," *Int. J. Comput. Vis.*, 2015, ISSN: 15731405. DOI: 10.1007/s11263-014-0788-3.

[23] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Jt. Conf. Neural Networks*, 2015, ISBN: 9781479919604. DOI: 10.1109/IJCNN.2015.7280696.

[24] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nat. Neurosci.*, 2006, ISSN: 10976256. DOI: 10.1038/nn1643.

[25] S. M. Bohte, H. L. Poutré, and J. N. Kok, "{SpikeProp}: Error-Backpropagation for Networks of Spiking Neurons," in *Proc. Eur. Symp. Artif. Neural Networks (ESANN 2000)*, 2000.

[26] F. Ponulak, "ReSuMe-new supervised learning method for Spiking Neural Networks," *Inst. Control Inf. Eng. Pozn. Univ.*, 2005, ISSN: 1530-888X.

[27] M. Stimberg, D. F. Goodman, V. Benichoux, and R. Brette, "Brian 2 - the second coming: spiking neural network simulation in Python with code generation," *BMC Neurosci.*, 2013, ISSN: 1471-2202. DOI: 10.1186/1471-2202-14-s1-p38.

[28] D. Purves, G. Augustine, D. Fitzpatrick, L. Katz, A.-S. LaMantia, J. McNamara, and M. Williams, *Neuroscience. 2nd edition: Autonomic Regulation of Cardiovascular Function*, 2001. arXiv: NBK10799.

[29] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE Trans. Audio, Speech Lang. Process.*, 2014, ISSN: 15587916. DOI: 10.1109/TASLP.2014.2339736.

[30] A. Tavanaei and A. S. Maida, "A spiking network that learns to extract spike signatures from speech signals," *Neurocomputing*, vol. 240, pp. 191–199, 2017, ISSN: 18728286. DOI: 10.1016/j.neucom.2017.01.088. arXiv: arXiv:1606.00802v3.

[31] M. Holmberg, D. Gelbart, U. Ramacher, and W. Hemmert, "Automatic speech recognition with neural spike trains," *9th Eur. Conf. Speech Commun. Technol.*, pp. 1253–1256, 2005.

[32] Z. Pan, J. Wu, M. Zhang, H. Li, and Y. Chua, "Neural Population Coding for Effective Temporal Classification," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2019-July, 2019. DOI: 10.1109/IJCNN.2019.8851858. arXiv: 1909.08018.

[33] Z. Pan, Y. Chua, J. Wu, M. Zhang, H. Li, and E. Ambikairajah, "An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks," *Front. Neurosci.*, vol. 13, pp. 1–23, 2020, ISSN: 1662453X. DOI: 10.3389/fnins.2019.01420. arXiv: 1909.01302.

[34] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Front. Neurosci.*, vol. 12, no. NOV, pp. 1–17, 2018, ISSN: 1662453X. DOI: 10.3389/fnins.2018.00836.

[35] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, 1997, ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[36] F. Chollet, "Keras: The Python Deep Learning library," *Keras.Io*, 2015.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, 2012, ISBN: 9781627480031. DOI: 10.1061/(ASCE)GT.1943-5606.0001284.

[38] A. Graves, *Supervised Sequence Labeling with Recurrent Neural Networks*. 2013, ISBN: 2000201075. DOI: 10.1145/2661829.2661935. arXiv: arXiv:1308.0850v1.

[39] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Precise-Spike-Driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns," *PLoS One*, 2013, ISSN: 19326203. DOI: 10.1371/journal.pone.0078318.

[40] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, "Jakobovski/free-spoken-digit-dataset: v1.0.8," Aug. 2018. DOI: 10.5281/ZENODO.1342401. [Online]. Available: https://zenodo.org/record/1342401.

[41] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," Apr. 2018. arXiv: 1804.03209. [Online]. Available: http://arxiv.org/abs/1804.03209.

[42] T. Rumbell, S. L. Denham, and T. Wennekers, "A spiking self-organizing map combining sTDP, oscillations, and continuous learning," *IEEE Trans. Neural Networks Learn. Syst.*, 2014, ISSN: 21622388. DOI: 10.1109/TNNLS.2013.2283140.

[43] H. Hazan, D. Saunders, D. T. Sanghavi, H. Siegelmann, and R. Kozma, "Unsupervised Learning with Self-Organizing Spiking Neural Networks," in *Proc. Int. Jt. Conf. Neural Networks*, 2018, ISBN: 9781509060146. DOI: 10.1109/IJCNN.2018.8489673. arXiv: 1807.09374.

[44] M. V. Wezel, "A robust modular spiking neural networks training methodology for time-series datasets,"

Parameters settings

A

A.1 Cochlear filter bank parameters

Cochlear filter index	Lower cut-off frequency (Hz)	Higher cut-off frequency (Hz)
1	106.78	254.21
2	177.40	337.74
3	254.21	428.59
4	337.74	527.38
5	428.59	634.83
6	527.38	751.67
7	634.83	878.75
8	751.67	1016.96
9	878.75	1167.26
10	1016.96	1330.71
11	1167.26	1508.48
12	1330.71	1701.81
13	1508.48	1912.06
14	1701.81	2140.72
15	1912.06	2389.39
16	2140.72	2659.84
17	2389.39	2953.95
18	2659.84	3273.82
19	2953.95	3621.68
20	3273.82	4000.00

Table A.1: The 20-channel cochlear filter bank parameters used. The lower cut-off frequency and higher cut-off frequency of each band-pass filter are listed.

A.2 Leaky integrate-and-fire model settings

Fixed parameter values	
C	100f
R	1G
Tref	0m
Vthresh	400m
Ibias	0

Table A.2: Leaky integrate-and-fire (LIF) model parameter settings used for training and testing network performance. These values correspond to the latest design of the neuromorphic hardware.

A.3 Tempotron parameter settings

Parameter	Value
Time step	1u
Learning rate	1e-17; 1e-18
V_rest	0
τ	1E-4
τ_s	3.3E-7
Threshold	4E-14

Table A.3: Parameters settings used to train the spiking parts of the architecture. Where $\tau = RC$ and τ_s is set close to zero, this since the synaptic time constant is not part of neuron model in the simulator. The settings for the Threshold much lower than the LIF neuron settings, as otherwise the output spike activity is too strong. The set value is found iteratively.

Classification results

B

B.1 Network size testing

B.1.1 Number of thresholds for PTA-SNN architecture (Clean)

# Thresholds	Input size*	Accuracy [%]	Bad unique class [%]	Avg. # spikes per sample
3	120	34.00	23.00	47.61
6	240	74.50	14.00	160.61
9	360	81.00	12.25	264.28
12	480	84.00	10.50	364.08
15	600	86.50	7.75	464.23
18	720	85.75	8.00	563.95
21	840	86.25	8.00	663.96

Table B.1: Raw testing results, for a variety of set number of thresholds. Input size is the number of required input-layer neurons of the SNN network. *Meaning the size of the input layer of the SNN architecture, output is fixed to corresponding number of classes.

B.1.2 Dimensions for SOM-SNN architecture (Clean)

SOM dimensions	Input size*	Accuracy [%]	Bad unique class [%]	Avg. # spikes per sample
2x2	4	26.25	35.5	426.68
4x4	16	74.00	15.75	426.53
6x6	36	81.75	9.75	425.13
8x8	64	85.75	8.50	427.04
10x10	100	90.25	7.25	427.24
12x12	144	93.00	3.75	425.99
14x14	196	94.00	3.50	425.73
16x16	256	95.25	2.25	424.87
18x18	324	97	2.25	424.14
20x20	400	96.75	2.75	421.74
22x22	484	96.25	3	419.79
24x24	576	97.5	1.25	418.46

Table B.2: Raw testing results, for a variety of SOM dimensions. Input size is the number of required input-layer neurons of the SNN network. *Meaning the size of the input layer of the SNN architecture, output is fixed to corresponding number of classes

B.2 SNR noise testing

B.2.1 PTA-SNN architectures

Architecture	Network structure	Accuracy [%]	Bad unique class [%]	Avg. # spikes per sample
P-TA15 CLEAN	600;10	86.00	7.25	464.24
P-TA15 CLEAN	600;10	86.5	7.75	464.23
P-TA15 SNR 20 MM	600;10	68.00	21.00	487.24
P-TA15 SNR 20 MC	600;10	78.50	14.50	487.01
P-TA15 SNR 10 MM	600;10	47.75	36.50	529.88
P-TA15 SNR 10 MC	600;10	71.25	22.75	529.29
P-TA15 SNR 0 MM	600;10	26.25	69.00	628.74
P-TA15 SNR 0 MC	600;10	50.75	38.00	626.55
P-TA15 SNR -5 MM	600;10	15.00	81.75	668.43
P-TA15 SNR -5 MC	600;10	30.00	59.50	665.58
P-TA18 CLEAN	720;10	85.75	8.00	563.95
P-TA18 SNR 20 MM	720;10	67.75	21.25	592.41
P-TA18 SNR 20 MC	720;10	79.00	13.50	592.07
P-TA18 SNR 10 MM	720;10	49.5	35.75	644.07
P-TA18 SNR 10 MC	720;10	70.00	21.50	643.23
P-TA18 SNR 0 MM	720;10	24.25	71.00	762.15
P-TA18 SNR 0 MC	720;10	51.75	37.75	760.03
P-TA18 SNR -5 MM	720;10	13.75	83.00	810.63
P-TA18 SNR -5 MC	720;10	29.25	57.00	807.78
P-TA21 CLEAN	840;10	86.25	8.00	663.96
P-TA21 CLEAN	840;10	87.00	7.75	663.93
P-TA21 SNR 20 MM	840;10	68.75	20.75	696.78
P-TA21 SNR 20 MC	840;10	78.00	14.00	696.6
P-TA21 SNR 10 MM	840;10	51.50	32.50	756.36
P-TA21 SNR 10 MC	840;10	69.75	21.25	755.89
P-TA21 SNR 0 MM	840;10	28.35	67.50	897.37
P-TA21 SNR 0 MC	840;10	49.50	41.00	895.68
P-TA21 SNR -5 MM	840;10	14.50	83.50	951.85
P-TA21 SNR -5 MC	840;10	32.25	55.75	949.55

Table B.3: Raw SNR noise testing results, for a select number of promising PTA-SNN architectures.

B.2.2 SOM-SNN architectures

Architecture	Network structure	Accuracy [%]	Bad unique class [%]	Avg. # spikes per sample
SOM12x12 CLEAN	144;144;10	93.00	3.75	427.61
SOM12x12 SNR 20 MM	144;144;10	71.25	19.75	427.55
SOM12x12 SNR 20 MC	144;144;10	92.00	4.75	427.6
SOM12x12 SNR 10 MM	144;144;10	49.00	35.75	425
SOM12x12 SNR 10 MC	144;144;10	84.50	10.25	430.1
SOM12x12 SNR 0 MM	144;144;10	23.50	58.75	421.5
SOM12x12 SNR 0 MC	144;144;10	57.75	29.50	435.73
SOM12x12 SNR -5 MM	144;144;10	13.75	64.50	422.49
SOM12x12 SNR -5 MC	144;144;10	23.75	50.00	444.21
SOM18x18 CLEAN	324;324;10	97	1.25	423.58
SOM18x18 SNR 20 MM	324;324;10	79	15	424.31
SOM18x18 SNR 20 MC	324;324;10	93.5	4	425.68
SOM18x18 SNR 10 MM	324;324;10	51.75	34.75	419.54
SOM18x18 SNR 10 MC	324;324;10	87.75	7.75	431.48
SOM18x18 SNR 0 MM	324;324;10	21.75	58.75	415.35
SOM18x18 SNR 0 MC	324;324;10	61.25	31.5	427.45
SOM18x18 SNR -5 MM	324;324;10	14.25	73.25	414.82
SOM18x18 SNR -5 MC	324;324;10	32.5	43.00	447.03
SOM20x20 CLEAN	400;400;10	96.75	2.75	421.74
SOM20x20 SNR 20 MM	400;400;10	86	9.5	420.01
SOM20x20 SNR 20 MC	400;400;10	95	2.75	423.95
SOM20x20 SNR 10 MM	400;400;10	58.5	30.25	415.66
SOM20x20 SNR 10 MC	400;400;10	87	7.75	429
SOM20x20 SNR 0 MM	400;400;10	23	52.25	412.86
SOM20x20 SNR 0 MC	400;400;10	62.25	25	440.53
SOM20x20 SNR -5 MM	400;400;10	13.5	65	412.01
SOM20x20 SNR -5 MC	400;400;10	35.25	40.25	447.12
SOM22x22 CLEAN	484;484;10	96.25	3	419.8
SOM22x22 SNR 20 MM	484;484;10	86.75	7.25	417.97
SOM22x22 SNR 20 MC	484;484;10	94.5	3.5	422.83
SOM22x22 SNR 10 MM	484;484;10	58.75	28	413.83
SOM22x22 SNR 10 MC	484;484;10	88.25	6.75	429.84
SOM22x22 SNR 0 MM	484;484;10	27.25	54.5	411.31
SOM22x22 SNR 0 MC	484;484;10	63.75	21.25	440.78
SOM22x22 SNR -5 MM	484;484;10	14.25	70.25	410.28
SOM22x22 SNR -5 MC	484;484;10	34	40.75	448.65
SOM24x24 CLEAN	576;576;10	97.5	1.25	418.46
SOM24x24 SNR 20 MM	576;576;10	87	8.25	416.67
SOM24x24 SNR 20 MC	576;576;10	95.25	3.75	422.52
SOM24x24 SNR 10 MM	576;576;10	59	25.75	385.48
SOM24x24 SNR 10 MC	576;576;10	89	7	428.54
SOM24x24 SNR 0 MM	576;576;10	26.75	55.25	408.16
SOM24x24 SNR 0 MC	576;576;10	63.25	22.25	440.06
SOM24x24 SNR -5 MM	576;576;10	13.75	66.25	407.14
SOM24x24 SNR -5 MC	576;576;10	35	42	447.45

Table B.4: Raw SNR noise testing results, for a select number of promising SOM-SNN architectures.

B.3 Distinguish between speech and noise

B.3.1 PTA-SNN architecture with NO dedicated noise class (Noise only)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	**X**
0	0	4	0	0	31	5	0	0	0	0	**0**
1	0	0	3	0	31	5	0	0	0	0	**1**
2	0	3	0	0	29	3	0	3	0	0	**2**
3	1	2	1	0	26	7	0	3	0	0	**1**
4	0	0	1	0	35	3	0	2	0	0	**0**
5	1	0	0	0	32	5	0	2	0	0	**0**
6	0	0	2	0	33	3	0	2	0	0	**1**
7	0	1	0	0	36	2	0	1	0	0	**0**
8	0	0	0	0	37	2	0	1	0	0	**0**
9	0	2	1	0	31	5	0	1	0	0	**0**

Table B.5: Confusion matrix for the PTA-SNN architecture when only background noise is presented to the network. Since the network is trained to classify the speech samples, no response from the system is desired as any output activity corresponds to a speech class. However, this is not the case, as most samples results in misclassification of speech instead of noise.

B.3.2 PTA-SNN architecture with dedicated noise class (Noise only)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	**X**
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	**0**	**0**	**0**	**0**	**3**	**0**	**0**	**0**	**0**	**0**	**396**	**1**

Table B.6: Confusion matrix for the PTA-SNN architecture when only background noise is presented to the network. The added class 10 corresponds to the noise, and it highly detected.

B.3.3 PTA-SNN architecture with dedicated noise class (Speech only)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	**X**
0	**37**	0	0	0	0	0	0	0	0	1	0	0
1	0	**33**	0	0	0	1	0	0	0	1	0	0
2	0	0	**37**	2	0	0	0	0	0	0	0	0
3	0	0	0	**37**	0	0	1	0	0	2	0	0
4	0	0	0	0	**37**	0	0	0	0	0	0	0
5	0	2	0	0	0	**37**	0	0	0	1	0	0
6	0	0	1	6	0	0	**25**	1	1	0	0	0
7	0	0	0	0	0	0	0	**39**	0	0	0	0
8	0	0	0	2	0	0	2	0	**34**	0	0	0
9	1	0	0	0	0	1	0	0	0	**35**	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0

Table B.7: Confusion matrix for the PTA-SNN architecture when clean audio is presented to the network, with addition of the dedicated noise class.

B.3.4 SOM-SNN architecture with dedicated noise class (Noise only)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	**X**
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**400**	**0**

Table B.8: Confusion matrix for the SOM-SNN architecture when only background noise is presented to the network. The added class 10 corresponds to the noise, and it highly detected.

B.3.5 SOM-SNN architecture with dedicated noise class (Speech only)

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	**X**
0	**37**	1	1	1	0	0	0	0	0	0	0	0
1	0	**39**	0	0	0	0	0	1	0	0	0	0
2	0	0	**39**	0	0	0	1	0	0	0	0	0
3	1	0	0	**39**	0	0	1	0	1	0	0	0
4	0	0	0	0	**40**	0	0	0	0	0	0	0
5	0	2	0	0	0	**38**	0	0	0	0	0	0
6	0	0	0	1	0	0	**38**	0	2	0	0	0
7	0	0	0	0	0	0	1	**40**	0	0	0	0
8	0	0	0	1	0	0	0	1	**40**	0	0	0
9	0	2	0	0	0	0	0	0	0	**38**	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0

Table B.9: Confusion matrix for the SOM-SNN architecture when clean audio is presented to the network, with addition of the dedicated noise class.

Power and latency calculations

C.1 CNN

– Section removed due to confidentiality –

C.2 LSTM*

– Section removed due to confidentiality –

C.3 RNN*

– Section removed due to confidentiality –

C.4 SOM

– Section removed due to confidentiality –