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

CAS-2021-00

M.Sc. Thesis

A Spiking Neural Network classification architecture for spatial-temporal data processing

Roy Arriëns

Abstract

A big catalyst of the AI revolution has been Artificial Neural Networks (ANN), abstract computation models based on the biological neural networks in the brain. However, they require an immense amount of computational resources and power to configure and when deployed often are dependent on cloud resources to function. This makes ANNs less suitable for edge computing devices where all these resources are scares. Spiking Neural Networks (SNN) are a new generation of neural networks which process information via sparse discrete time events, called "spikes". When mapped to neuromorphic hardware, SNNs promise high energy efficiency and low computational latency.

This work proposes a SNN classification architecture, using it to classify radar based hand gesture signatures. Literature on this topic is limited, leading us to explore certain ANN topologies to test our assumptions. By considering additional design limitations, we aim to find a neuromorphic hardware compatible design. While the proposed architecture is still limited in terms of classification accuracy. Our experiments have exposed interesting relationships between network sizes, accuracy and dimensionality reduction in SNNs.

A Spiking Neural Network classification architecture for spatial-temporal data processing

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Engineering

by

Roy Arriëns
born in Rotterdam, 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 **"A Spiking Neural Network classification architecture for spatial-temporal data processing"** by **Roy Arriëns** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 24-08-2021

Chairman: _____
dr.ir. T.G.R.M. van Leuken

Advisor: _____
K. Kozdon

Committee Members: _____
dr. R. Bishnoi

_____

Abstract

A big catalyst of the AI revolution has been Artificial Neural Networks (ANN), abstract computation models based on the biological neural networks in the brain. However, they require an immense amount of computational resources and power to configure and when deployed often are dependent on cloud resources to function. This makes ANNs less suitable for edge computing devices where all these resources are scares. Spiking Neural Networks (SNN) are a new generation of neural networks which process information via sparse discrete time events, called "spikes". When mapped to neuromorphic hardware, SNNs promise high energy efficiency and low computational latency.

This work proposes a SNN classification architecture, using it to classify radar based hand gesture signatures. Literature on this topic is limited, leading us to explore certain ANN topologies to test our assumptions. By considering additional design limitations, we aim to find a neuromorphic hardware compatible design. While the proposed architecture is still limited in terms of classification accuracy. Our experiments have exposed interesting relationships between network sizes, accuracy and dimensionality reduction in SNNs.

Acknowledgments

First of all, I would like to thank my supervisor K. Kozdon for their incredible support over the course of the project. Without it, I would not have been able to achieve my goals. I aspire to be just as diligent, structured and hard working. I cannot express my gratitude enough.

Furthermore, I would like to thank dr.ir. T.G.R.M. van Leuken, dr. A. Zjajo and dr.ir. S.S. Kumar for giving me the opportunity to work on such an interesting and promising topic. It has opened my eyes to the challenges and enormous potential of Spiking Neural Networks.

To all the CAS group members and master students for creating the possibility to ask questions and have discussions every week. Specially, the weekly chats with the master students, which were a much appreciate break from the daily grind of the project.

To my family, who's unconditional love and support carried my through the tough times. My girlfriend Didi, who helped me get through the intense final weeks. To my good friends, Martijn & Jitske who were part of our so called "Lockdown club". The fun we had despite the pandemic has helped me stay on track and will remain with me forever as precious memories.

Finally, a special thanks to my late grandmother. Who sadly passed away this year and couldn't see me receive my degree. Her love and caring nature has made me the person who I am today.

To all of you,
Thank you!

Roy Arriëns
Delft, The Netherlands
24-08-2021

Contents

Abstract v

Acknowledgments vii

1 Introduction 1
1.1 Thesis objective . . . 1
1.2 Contributions . . . 2
1.3 Thesis outline . . . 2

2 Related Work 3

3 Background 5
3.1 Biological neuron . . . 5
3.2 Artificial Neural Networks . . . 7
3.3 Spiking Neural Networks . . . 9
3.4 Spiking neuron models . . . 10
3.4.1 Hodgkin-Huxley Model . . . 11
3.4.2 Leaky Integrate and Fire Model . . . 12
3.5 Spike encoding . . . 13
3.6 Training methods . . . 14
3.6.1 ANN-SNN conversion . . . 14
3.6.2 Supervised training: Tempotron . . . 14
3.6.3 Unsupervised training: Spike Timing Dependent Plasticity & Triplet Spike Timing Dependent Plasticity . . . 15
3.7 Self Organizing Map . . . 17
3.8 Spiking Self Organizing Map . . . 19
3.9 Radar & The Doppler effect . . . 20
3.10 Micro-Doppler effect . . . 21

4 Methods 23
4.1 Radar dataset . . . 23
4.2 SNN classifier . . . 25
4.3 Proposed architecture . . . 26
4.4 Encoder . . . 28
4.5 Decoder . . . 30
4.6 Baseline architectures . . . 31

5 Experiments & Results 35
5.1 Spiking Neural Network Simulator . . . 35
5.2 Experiment setup . . . 36
5.3 Baseline architectures experiments . . . 38
5.4 SNN classifier . . . 44

5.5 Self Organizing Map & SNN classifier 48
5.6 Spiking Self Organizing Map & SNN classifier 51

6 Conclusion **61**

7 Future Work **63**

A Appendix **71**
A.1 Experiment results LSTM classifier 71
A.2 Experiment results SOM-LSTM architecture 72
A.3 Experiments results SOM-SNN architecture 72

List of Figures

1.1 Hand gesture recognition using radar in Google's Soli project 2

3.1 Anatomy of a biological neuron 5
3.2 Multiple stages of a neurons membrane potential 6
3.3 Signal transmission in the synapse 6
3.4 The Multilayer perceptron 7
3.5 The Perceptron 8
3.6 The SNN classifier pipeline 9
3.7 The spiking neuron 10
3.8 Neuron models biological plausibility vs implementation cost 10
3.9 The electrical circuit of the Hodgkin-Huxley model 11
3.10 The electrical circuit of the Leaky Integrate and Fire model 12
3.11 Rate based vs temporal based encoding 13
3.12 The long term depression(LTD) and long term potentiation(LTP) pairs 15
3.13 The Self Organizing Map 17
3.14 Clustering example: MNIST images in a 25x25 grid 18
3.15 A spiking self organizing map structure 19
3.16 A basic radar system 20
3.17 The Micro-Doppler signature of a simulated helicopter 21

4.1 The handgestures in DopNet from l.t.r Wave, Click, Pinch & Swipe . . 23
4.2 The Micro-Doppler signatures of the four gestures; Wave, Click, Pinch and Swipe 24
4.3 The SNN classifier 25
4.4 The complete architecture with SSOM extention 26
4.5 The population threshold algorithm encoding (a) a signal with 10 thresholds (b) the resulting spike trains with 20 encoding neurons 28
4.6 The gesture wave and swipe encoded with PTE using 5 thresholds. Here a subset of the data was taken from 100 Hz to-100 Hz 29
4.7 An output layer with four neurons and a First-time-to-spike decoder . . 30
4.8 A LSTM cell 31
4.9 The baseline classifier using Long Term Short Memory cells 32
4.10 The extension of the baseline classifier with a Self Organizing Map . . . 33

5.1 The class distribution in the DopNet dataset, (a) original (b) after random sampling 36
5.2 The separation of a dataset using 5-fold cross validation 37
5.3 10-fold stratified cross validation classification accuracy for varying LSTM sizes. For an numerical overview of the results see Table A.1. . . 38
5.4 Confusion matrices using fold 4 as test set with LSTM sizes (a) 2 cells (b) 12 cells (c) 32 cells 40

5.5 10-fold stratified cross validation classification accuracy for varying amount of neurons in the SOM, with LSTM size 32. For an numerical overview of the results see Table A.2 42
5.6 Confusion matrices using fold 4 as test set with SOM grid sizes (a) 4x4 grid (b) 10x10 grid (c) 16x16 grid 43
5.7 The four feature selection scenarios (a) -200 Hz to 200 Hz (b) -100 Hz to 100 Hz (c) -50 Hz to 50 Hz (d) -25 Hz to 25 Hz 45
5.8 SNN 10-fold stratified cross validation classification accuracy for varying SOM grid sizes.For an numerical overview of the results see Table A.3 . 49
5.9 SNN feature selected 10-fold stratified cross validation classification accuracy for varying SOM grid sizes. The error bars are omitted for visualization purposes. For an numerical overview of the results see Tables A.3 to A.7 50
5.10 The classification accuracy and E_{MDS} for sweeps of the lower(U_{min}) and upper(U_{max}) bound of the initial weights 54
5.11 The classification accuracy and E_{MDS} for sweeps of A_{2-} and A_{2+} . . . 55
5.12 The classification accuracy and E_{MDS} for sweeps of τ_+ and τ_- 56
5.13 The classification accuracy and E_{MDS} for sweeps of the lower(U_{min}) and upper(U_{max}) bound of the initial weights, now with higher weight values 57
5.14 The classification accuracy and E_{MDS} for sweeps of neighbourhood function shapes 58

List of Tables

4.1 State of the art accuracies for DopNet at the early stages of the project 31

5.1 Parameters for the LSTM classifier 38
5.2 Parameters for the SOM-LSTM architecture 41
5.3 Parameters for the Tempotron learning method. The synaptic weights are initialized following a uniform distribution between -1 fC and 1 fC, here the uniform distribution is denoted as U. Also, τ_s is set close to zero as the simulator assumes spikes are modelled as instantaneous events . 44
5.4 Amount of input neurons for thresholds 44
5.5 The 10-fold classification accuracy and input layer sizes when performing feature selection from -200 Hz to 200 Hz and -100 Hz to 100 Hz 46
5.6 The 10-fold classification accuracy and input layer sizes when performing feature selection from -50 Hz to 50 Hz and -25 Hz to 25 Hz 46
5.7 The confusion matrix of 6 thresholds with all features of fold 4. Here the X column denotes a no response 47
5.8 The confusion matrix of 6 thresholds with features between -100Hz and 100Hz of fold 4. Here the X column denotes a no response 47
5.9 The experiment results for fold 4 running for longer epochs, with features selected between -100 Hz and 100 Hz and using 6 thresholds 47
5.10 Parameters for the SOM-SNN architecture. The synaptic weights are initialized following a uniform distribution between -1 fC and 1 fC, here the uniform distribution is denoted as U 48
5.11 The confusion matrix of a 18x18 SOM grid for fold 4. Here the X column denotes a no response 48
5.12 Assumed parameters of the input encoder 51
5.13 Parameters for the SNN classifier trained by Tempotron, identical to the one used in section 5.4 52
5.14 The baseline parameters for the SSOM at the start of the grid search. . 53
5.15 The final configuration for the SSOM found after the grid search 59
5.16 The confusion matrix of the SSOM-SNN architecture after the grid search and using fold 4 as the test set. Here the X column denotes a no response 59

A.1 10-fold cross validation classification accuracy of the LSTM classifier for varying LSTM sizes 71
A.2 10-fold cross validation classification accuracy of the SOM-LSTM architecture for varying SOM grid sizes 72
A.3 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using all features 72
A.4 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -200 Hz and 200 Hz 72

A.5 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -100 Hz and 100 Hz 73
A.6 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -50 Hz and 50 Hz 73
A.7 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -25 Hz and 25 Hz 73

1 Introduction

In recent decades huge steps have been made in the development of AI technology, specifically Artificial Neural Networks (ANNs) due their ability to effectively solve complex non-linear computing problems. These networks have benefited from the advancement of the classical Von Neumann computer architecture. Using accelerators like GPUs to speed up training clears the way for even larger and complex networks. The interest in solving these problems has sparked the development of popular ANN software libraries like Tensorflow [1] and Pytorch [2], enabling for more widespread adoption of this technology.

Spiking neural networks (SNNs) are a new emerging type of Neural Network, more biologically focused and processes information using discrete events called "spikes". SNNs can be mapped onto specialized neuromorphic hardware, promising a significant higher energy efficiency with respect to ANNs. Not many standard standard software libraries for designing and simulating SNNs exist, but interest has been increasing in recent years. This is mainly due to a rise in the amount of research done in non Von Neumann hardware architectures, specifically designed for SNNs.

1.1 Thesis objective

Due to an increasing amount of electrical devices users are interfacing with, faster and easier control of these devices is desired. Hand gestures is one of the most natural way for humans to communicate, it does not require any understanding of language interpretation or verbal commands. A recognition system is needed to convey the information stored in a hand gesture to the respective computing platform. Such a system often consists of a sensor and some computational logic. Hand gesture recognition can be done with visual sensors like cameras, however this can raise concerns for privacy. Also, light conditions and other environmental settings can affect the performance of a visual based sensor. Radar(Figure 1.1) on the other hand does not suffer from these problems and due to the increasingly scaling down of radar hardware, it becomes viable in low power applications. These devices are often battery powered, which imposes constraints on the power consumption of the computing hardware. SNNs can solve the same complex non linear problems, normally solved by ANNs using only a fraction of the energy. Making them excellent candidates for devices with strict power budgets.

Currently there exist little literature about SNN architectures which can classify radar based hand gestures which takes into account hardware compatibility. This is mainly due to the fact neuromorphic hardware development is still in it's early stages and not yet widespread available.

The goal of this thesis project will be to develop a SNN architecture which can classify hand gestures using data from a radar, while taking into account.

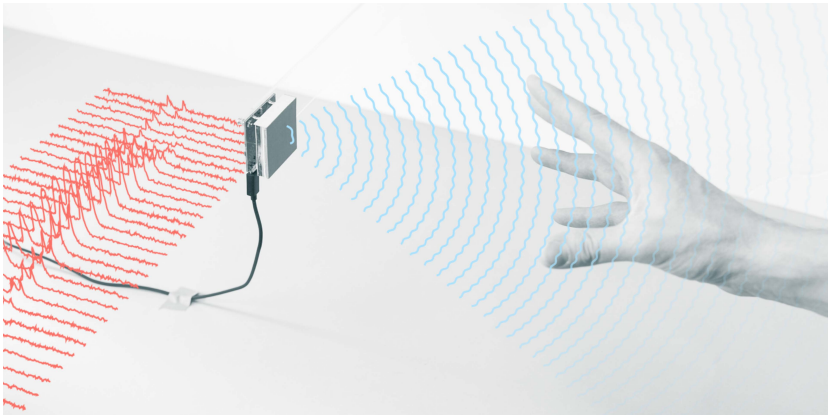


Figure 1.1: Hand gesture recognition using radar in Google's Soli project[3]

1.2 Contributions

The contributions of this thesis project are:

- A Python tool for pre-processing the DopNet[4] Micro-Doppler dataset
- An ANN baseline network which can classify Micro-Doppler hand gesture signatures implemented using Tensorflow
- Analysis of the DopNet dataset
- A SNN classifier based on the Tempotron [5] learning rule
- A SNN based feature reduction mechanism
- Guidelines on parameter selection for SNN and ANN networks.
- A Python tool to calculate dissimilarity metrics between spike encoded samples
- Made numerous contributions to an SNN simulator in order to support experiments

1.3 Thesis outline

This thesis report is structured as follows. Chapter 2 discusses the state of the art on radar-based hand gesture recognition and SNNs. In chapter 3 relevant background information on methods and algorithms is presented. Chapter 4 discusses how the proposed architecture and which design decisions were made. The simulation environment and the simulation results are discussed in chapter 5. Chapter 6 concludes on the findings of the report. Recommendations on possible future improvements are given in 7.

2 Related Work

Only recently has there been published work on radar based hand gesture recognition with Spiking Neural Networks. In [6] a hybrid SNN-ANN architecture is proposed able to classify Range-Doppler images and Micro-Doppler signatures of hand gestures. A Liquid State Machine(LSM) is employed to map the input spikes to a state within the LSM. The state is readout by a ANN classifier that predicts the class of the input. The proposed method achieves state-of-the-art results, with 98% on both types of data.

A recurrent SNN is proposed in [7] able to perform classification on ECG, audio and Range-Doppler data. Multiple layers of recurrent connected spiking neurons are trained by a novel surrogate-gradient descent method referred to as "Multi-Gaussian", based on backpropogation-through-time (BPTT) algorithm. The authors show the effectiveness of the proposed network, while providing a theoretical energy consumption estimations.

The classification of human actions is done by [8], with a Convolutional Spiking Layer and a logistic regression classifier. Micro-Doppler frames are greyscale and binarized as part of the preprocessing stage. Multiple layers of spiking neurons inspired by Convolutional Neural Networks (CNNs) are used to extract the spatial features.

ANN based solutions have been a research topic for some years now. A common approach is to view the radar data as an image problem and use CNNs as feature extractors in combination with a classifier network. The authors in [9, 10] split the processing in a CNN feature extraction layer and temporal classification using Long Short Term Memory (LSTM) on Range-Doppler frames. A similar approach is presented in [11], who use a multilayer perceptron (MLP) in the classification layer instead, classifying range-time data format.

A novel combination of CNN & Temporal CNN (TCN) is proposed in [12]. Capable of reaching state-of-the-art in terms of accuracy, while also showing the network can run on embedded hardware with an system-level power consumption of 120 mW. Another work [13] has show based hand gesture recognition, running a CNN on a Raspberry Pi and Intel's Neural Compute Stick 2; a hardware accelerator for deep neural network algorithms. While training is still dependent on cloud resources, their proposed network achieves worst case scenario 97,50% on these embedded platforms.

Other solutions tend to focus on using a machine learning models trained on hand picked features. Instead of focusing on tiny movements in hand gestures, [14] assumes large periodic movements e.g. circular and repeated translational motions. From the recorded radar signal the features;Doppler frequency, spectrum contrast, spatial dispersion and gesture validity are extracted to train a decision tree classifier. Tested classification accuracy ranges from 95% to 99% and execution latency is $\sim$ 1 ms

Background 3

3.1 Biological neuron

The biological neuron or nerve cell is seen as the fundamental biological unit in the nervous system of almost all animals. Responsible for the processing and transmitting of information throughout the nervous system. Neurons are interconnected via synapses to neighbouring neurons forming large-scale brain network, performing specific cognitive tasks e.g visual processing or motor control.

A biological neuron consists of three main parts; the cell body also referred to as the soma, the dendrites and the axon. The dendrites can be seen as the input to the cell body, receiving electrical signal from the neighbouring neurons. In the soma these signals are processed determining the neuron's state. It's equivalent to an analog memory element, as it contains information about the previous seen signals at the dendrites. However this information will slowly leak away as the neuron will return to it's resting state. Depending on it's state, the neuron can send out an electrical signal across it's axon to the other neurons.

Neurons can vary in shape and size based on their specific function. In this thesis, only multipolar neurons are assumed, with a single axon and multiple dendrites illustrated in Figure 3.1.

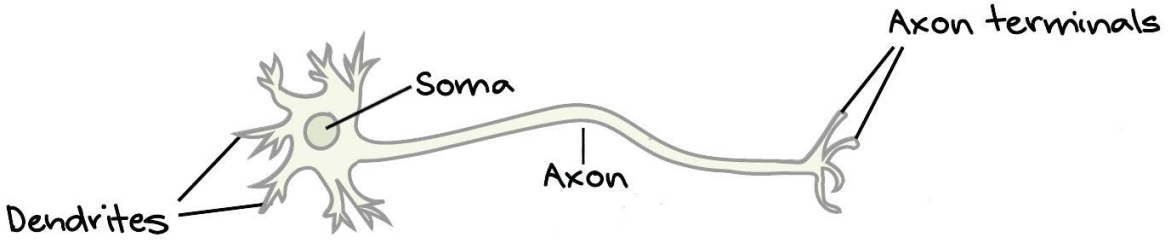


Figure 3.1: Anatomy of a biological neuron [15]

The nucleus is enclosed by a plasma membrane also called the neuron membrane for protection. Across the membrane a electrical potential(voltage) exists, which is influenced by the electrochemical signals send out by neighbouring neurons, arriving at the dendrites. If there's sudden change in the membrane potential an action potential or *spike* is send across the axon towards the synaptic terminals.

Figure 3.2 illustrates the different stages of the membrane potential. At first the membrane is in its resting potential at -70 mV. Input stimuli start arriving at the neuron's dendrites, if the incoming stimuli don't carry enough energy the membrane potential will not reach its firing threshold(failed initiations) and over time the energy will leak away and the potential will return to its resting voltage. If the membrane

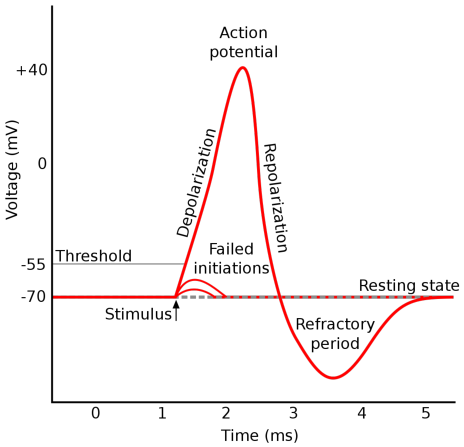


Figure 3.2: Multiple stages of a neurons membrane potential [16]

does cross the firing threshold, an action potential which will be send the neighbouring neurons.

After the action potential has been elicited, the neuron will enter a refractory period where it's membrane potential decreases below it's resting voltage, leaving the neuron unable to create a new action potential. Afterwards it will return to it's resting voltage.

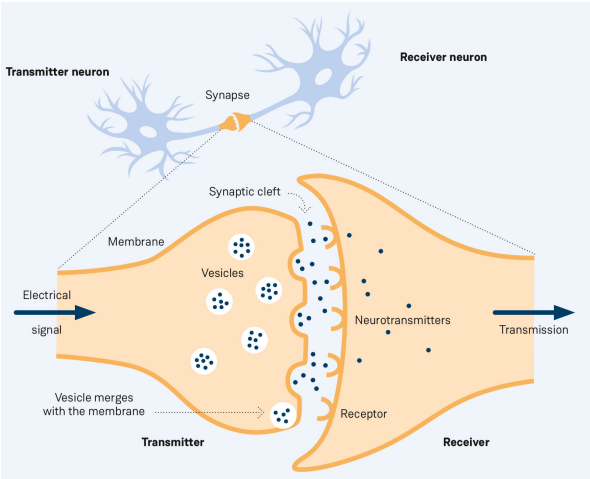


Figure 3.3: Signal transmission in the synapse [17]

Travelling along the axon, the action potential reaches the synaptic terminals. There exists two different types of synapses. Electrical synapses connect the presynaptic and postsynaptic membrane via *gap junctions* capable of carrying electrical current for fast signal transmission. Chemical synapses converts the action potential into neurotransmitters and are released into the synaptic cleft, further illustrated in Figure 3.3. Some neurotransmitters are bound to receptors on the postsynaptic side completing the transmission.

3.2 Artificial Neural Networks

Artificial neural networks (ANNs) are abstract computational models inspired by biological neural networks. The widespread success of these networks is due to their ability to perform non-linear tasks and adapt or *learn* on a large variety of data. These tasks vary from data processing, regression and classification. For the context of this thesis only classification will be discussed.

Classification is the process of assigning a class label to a set of new observations based on previous seen data. A *classifier* can be seen as mapping function f:

$$\hat{y} = f(\mathbf{x}; \mathbf{W}) \tag{3.1}$$

with $\hat{y}$, $\mathbf{x}$ and $\mathbf{W}$, the predicted class label, the observation data and the tunable parameters of the classifier. By tuning $\mathbf{W}$ in a process called "training", a mapping can be found. To test the quality of the mapping, validation data is used to asses the performance based on one or more metrics.

A classical feedforward ANN is the multilayer perceptron (MLP), a network of *perceptrons*[18] organized in different layers, illustrated in Figure 3.4. The network consist of an input Layer, one or more hidden Layer(s) and a output layer. The terminology *hidden* is used since these layers cannot be accessed by external systems or users.

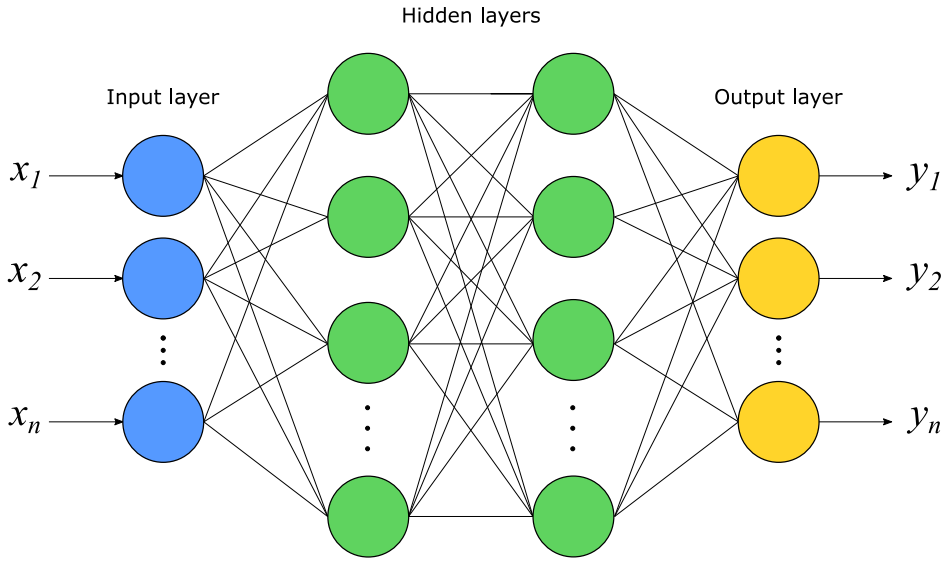


Figure 3.4: The Multilayer perceptron

The perceptrons are interconnected via weighted edges, similar to synapses in biological neurons. A perceptron (Figure 3.5) has a set of inputs $x_1, x_2 ... x_n$, that are mapped to an activation o_j:

$$o = \varphi(\mathbf{w}\mathbf{x} + b) \tag{3.2}$$

where $\mathbf{w} \ \epsilon \ \mathbb{R}^m$ is the perceptron's weight vector, b_j a bias term and activation function φ. Common activation functions are the tanh and sigmoid function, but recently the softmax and ReLu have gained in popularity in the deep learning field.

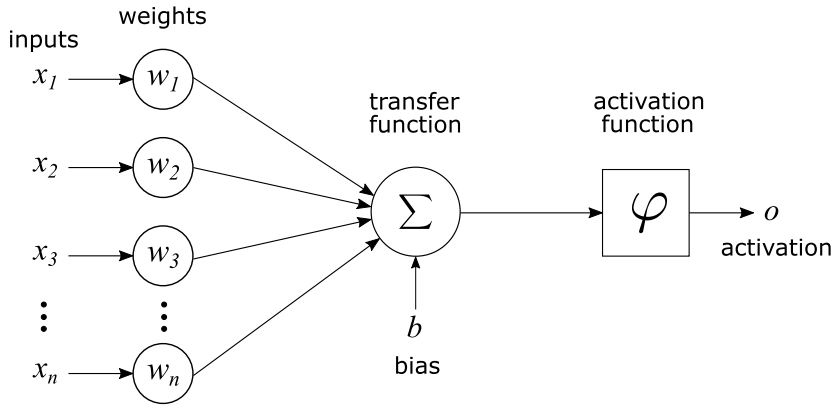


Figure 3.5: The Perceptron

A common algorithm used to train MLPs and ANNs in general is the Backpropagation [19] algorithm. Classical backpropagation minimizes the means squared error (MSE) between the target class y and the predicted class $\hat{y}$:

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3.3}$$

Gradient descent is applied to the gradient of MSE error with respect to the weights of the network ($\mathbf{W}$). The gradient calculation start at the output layer and is backwards propagated through the network, reusing the gradient calculation from the previous layer.

A major challenge when training an ANN is the concept of overfitting. This occurs when a model fits too much to its training data, effectively memorizing it. When a model is trained to long or too complex for its target data it can learn irrelevant information and relationships in the training data. When this happens, the model is unable to generalise on unseen data. It can be detected by using a part of the dataset as a separate test set. If performance on the training set will is high and the test set has a very low performance, it's likely due to overfitting. Various techniques exists to lessen the amount of overfitting e.g. cross-validation, regularization, pruning, etc..

3.3 Spiking Neural Networks

Spiking neural networks (SNNs) aims to solve the same task that are normally solved with ANNs. Information is propagated through these networks via discrete time events or *spikes*, equivalent to the action potentials discussed in section 3.1. Here we assume the communication between neurons to be solely based on the presence or absence of a spike. Meaning we don't account for any dynamics caused by neurotransmitters or receptors.

A typical SNN is a group of spiking neurons interconnected via *synapses*. In order for such networks to perform tasks on numerical data, an encoding into sequences of spikes via a spike encoder is required. On the other end, the output needs to be decoded in an interpretable format. The general pipeline for a SNN classifier is shown in Figure 3.6.

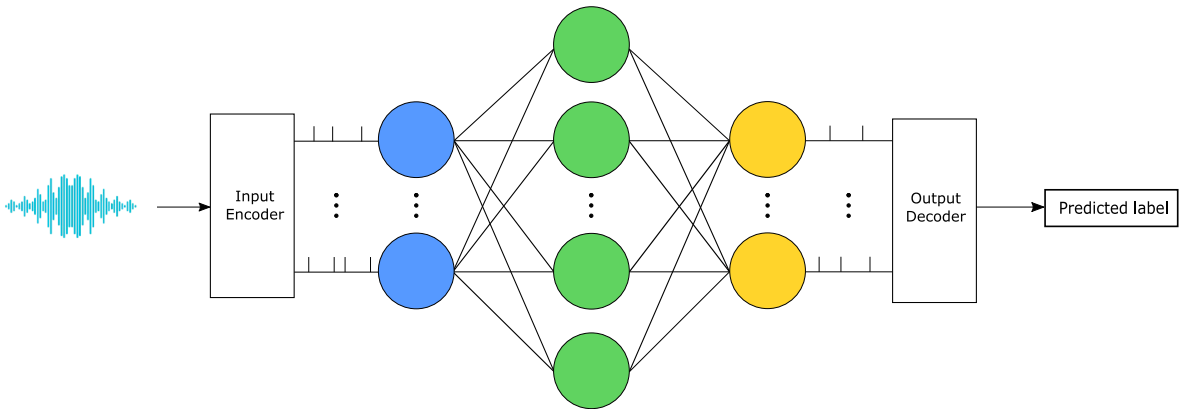


Figure 3.6: The SNN classifier pipeline

With the introduction of the time domain eq. (3.1) changes into:

$$\hat{y(t)} = f(\mathbf{x(t)}; \mathbf{W}) \tag{3.4}$$

where $\mathbf{y(t)}$ are the output spike trains, $\mathbf{x(t)}$ the input spike trains and $\mathbf{W}$ the synaptic weights or efficacies given as units of electrical charge in Coulombs (C).

Whereas neurons in feedforward ANNs propagates information only during each transmission cycle, a spiking neuron will transmit or *fire* whenever its electrical membrane potential reaches a certain threshold.

Figure 3.7 shows a spiking neuron where $X_1(t), X_2(t)...X_k(t)$, $V(t)$ and $S(t)$ are the input spikes trains, membrane potential and output spike train respectively. The firing dynamics can be written as:

$$I(t) = \sum_{i=1}^{n} x_i(t) w_i \tag{3.5}$$

$$S(t+1) = \begin{cases} 1, & \text{if } \varphi(I(t)) + V_m(t) \geq V_{th}. \\ 0, & \text{if } \varphi(I(t)) + V_m(t) < V_{th}. \end{cases} \tag{3.6}$$

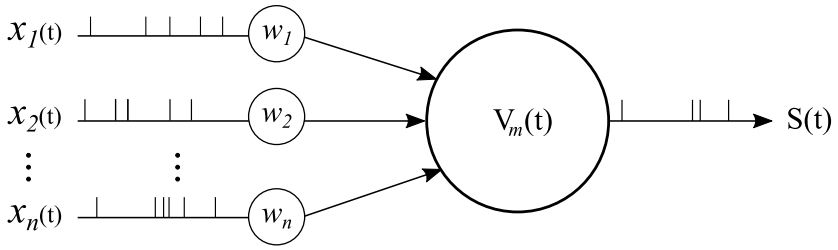


Figure 3.7: The spiking neuron

where $I(t)$ is the total incoming electrical charge from the input spike trains, V_{thr} the firing threshold and φ the model for inner dynamics of the neuron.

Previously in section 3.1, we discussed that the membrane potential basically acts as an analog memory element, as it retained information about it's previous inputs slowly leaking away. This can also be observed in Equation (3.5), where the current value of the membrane potential influences the firing behaviour. Each separate neuron can memorize information from it's recent input. This property is useful when processing temporal data, where correlations often express themselves over time.

3.4 Spiking neuron models

The complex electrochemical behaviour of biological neurons has a very high computational cost to reproduce. However, approximations can be made to capture the desired behaviour while keeping implementation cost within reasonable bounds. This does introduces the discussion of biological plausibility against implementation cost. Common neuron models often follow a trend, where higher biological plausibility comes at higher implementation cost, illustrated in Figure 3.8. However, there are exceptions to this trend.

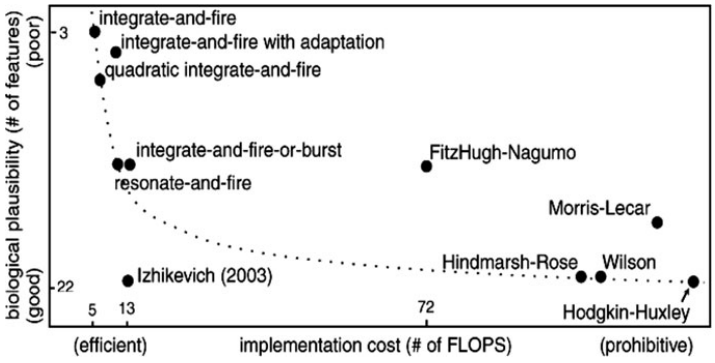


Figure 3.8: Neuron models biological plausibility vs implementation cost [20]

3.4.1 Hodgkin-Huxley Model

The Hodgkin-Huxley model [21] is the classical neuron model first described by Hodgkin & Huxley in 1952. The model was found by examining the dynamics of action potentials in the squid giant axon. The membrane is modelled as a capacitance C_M, illustrated in Figure 3.9. Current can flow through the membrane by charging C_M or via the ionic current channels in parallel with the capacity. This ionic current is divided into I_{Na} and I_K, which describe the sodium and potassium ions in the membrane and a leakage current I_l, describing other ions in the membrane.

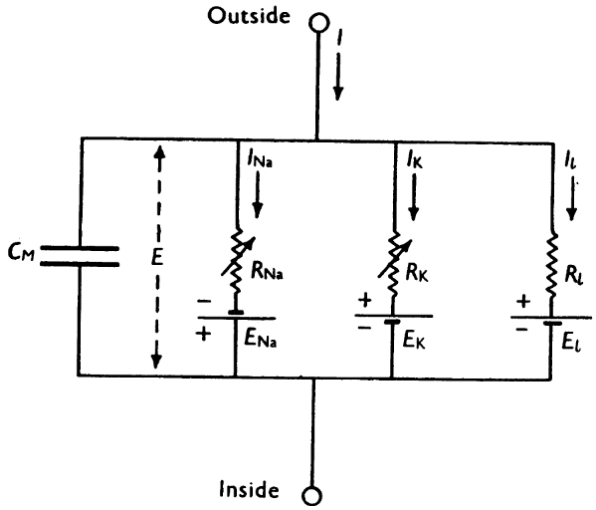


Figure 3.9: The electrical circuit of the Hodgkin-Huxley model [21]

The total membrane current is given by:

$$I = C_M \frac{dV}{dt} + I_{Na} + I_K + I_l \tag{3.7}$$

In their experiments Hodgkin and Huxley found that I_{Na} and I_K are defined by a set of non-linear differential equations. They can't be solved analytically and require a numerical approach. To simulate the dynamics of a single neuron will requires a decent amount of computing power. When scaling up to network with hundreds or even thousands of neurons, this becomes unmaintainable. Also from a hardware point of view, this model requires multiple variable resistors and controllable voltage sources for the equilibrium potentials E_{Na}, E_x and E_l for a single neuron. Implementing this is hardware for larger network wouldn't be cost effective as it would require enormous amount of resources. Therefore a simpler analytically solvable model is desired, with a feasible hardware design even for larger networks.

3.4.2 Leaky Integrate and Fire Model

The leaky integrate-and-fire model (LIF) adds a leaky term to the integrate-and-fire model [22, 23] proposed by Louis Lapicque. This model is a electrical RC circuit, where the membrane is modelled as a constant value capacitor C_M with a leaky resistor R_{leak}, illustrated inFigure 3.10.

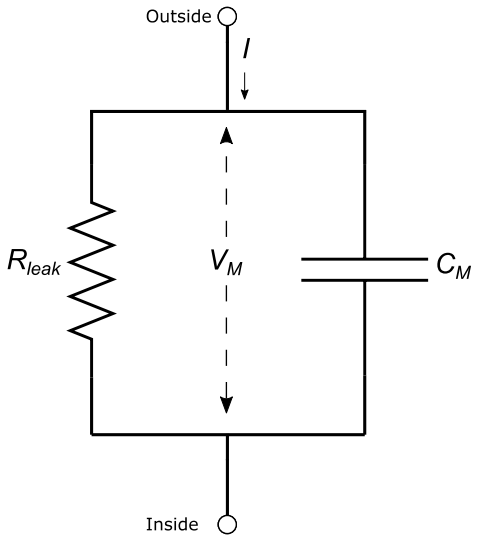


Figure 3.10: The electrical circuit of the Leaky Integrate and Fire model [24]

The dynamics of this model can be described by:

$$C_M \frac{dV_M(t)}{dt} = I(t) - \frac{V_M(t)}{R_{leak}} \tag{3.8}$$

$$V_M(t+1) = \begin{cases} V_M(t) + \frac{dt}{C_M}(I(t) - \frac{V_M(t)}{R_{leak}}) & \text{if } V_M(t) < V_{th} \\ V_{rest} & \text{if } V_M(t) \geq V_{th} \end{cases} \tag{3.9}$$

where V_M is the membrane voltage, I_{input} is the input current, R_{leak} is the leaky resistor and C_M the membrane capacitance.

The membrane will keep accumulating electrical charge while current I is flowing, at the same time charge is leaking away via R_{leak}. When $V_m(t)$ reaches the threshold voltage V_{th}, it will be set to it's resting voltage V_{rest}.

The LIF has replaced the complex non-linear differential equations seen in the Hodgkin-Huxley model with a solvable linear system. This reduces implementation cost dramatically at the cost of biological plausibility. In the context of SNNs this is an acceptable trade off. The goals isn't to make a computing system which is identical to the brain, rather is takes inspiration from biological neurons allowing for approximations.

Important to note that the models discussed until now don't take into account the actual morphology of a neuron. The biological neuron can have large wide spanning dendritic trees. Here we assume the neuron to be a computational unit, without any topology.

3.5 Spike encoding

As discussed in section 3.3 the numerical input data needs to be encoded into spatial temporal spike trains. Neurosciences has categorized in *rate-based* and *temporal-based*, often also referred as *spike-based*. Rate-based system encode information into an average amount of spikes over time, higher values are translated into higher firing rates. Temporal encoders on the other hand encode information in a sequence of spikes based on the precise timing of single spikes. The differences is illustrated in Figure 3.11. Temporal encoders are credited to be more energy efficient than rate encoders as they need less spikes to represent the same information. However, they are more susceptible to noise. Since a single noisy spike will carry more value in temporal encoder. Neural coding is an ongoing debate among neuroscientists as there's is no definitive answer whether biological neurons use rate-based or temporal based encoding. A point can be made that temporal encoding still contains rate information, this prompts the question whether rate based system are sufficient or precise time information is needed, as discussed in [25].

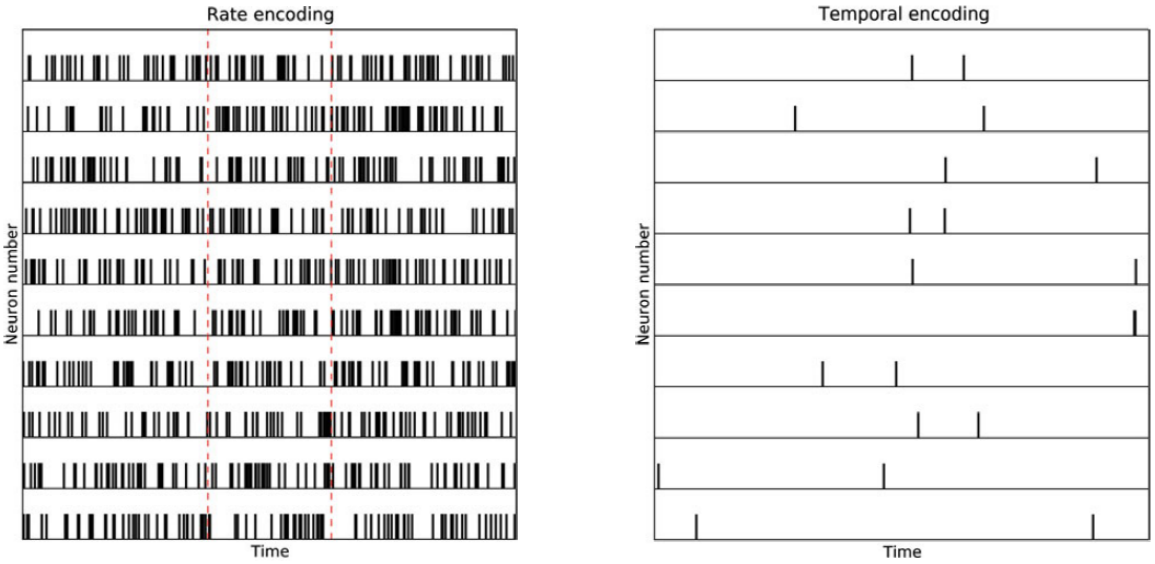


Figure 3.11: Rate based vs temporal based encoding [20]

3.6 Training methods

Most ANNs use the backpropagation algorithm to tune their weights and minimize the classification error. Unfortunately it's not possible to apply backpropagation directly to SNNs, because of two major differences with respect to ANNs. First, discrete spike events are non differentiable, meaning the gradient of the error cannot be calculated in the conventional way. Second, the temporal nature and leaky membrane potential makes finding the contributor to the error very difficult. At the time the error has been determined, neurons in earlier layers will already be in at their resting potential, making it unclear if they contributed to the error.

To circumvent this problem, different learning methods have been developed. We can categorize these learning methods in three different categories; ANN-SNN conversion, supervised and unsupervised methods. Supervised training makes use of labeled dataset. The label information is used to calculate the error of the prediction, which is then used to update any weights. Unsupervised methods on the other hand don't need this label information and determine the weight update on something else besides the prediction error. Each category will be discussed further with an example.

3.6.1 ANN-SNN conversion

A common method to create multi layered SNNs, is to train an ANN with backpropagation and then converting it into a SNN. According to [26], the basic principle of the conversion revolves around matching the activations in the ANN to firing rates of spiking neurons. The authors show that features from convolutional neural networks (CNN), like max pooling, softmax activation and batch normalization are compatible with SNNs.

3.6.2 Supervised training: Tempotron

The Tempotron [5] learning rule is a supervised method which allow neurons to learn on timing based spike patterns. The original paper assumes that neurons are modelled using the LIF neuron model driven by exponentially decaying synaptic currents:

$$V(t) = \sum_i w_i \sum_{t_i} K(t - t_i) + V_{rest} \tag{3.10}$$

Here V_{rest} is the resting voltage of the neuron, t_i are the spiking times coming from the ith input synapse and $K(t - t_i)$ is the incoming post synaptic spike. In our implementation $V(t)$ will be the LIF model, not the neuron model proposed in the original paper.

The incoming postsynaptic spikes are given by:

$$K(t - t_i) = \begin{cases} V_0(\exp[-(t - t_i)/\tau_m] - \exp[-(t - t_i)/\tau_s]) & t \leq t_i \\ 0 & t < t_i \end{cases} \tag{3.11}$$

Here V_0 is a normalization factor, τ_m and τ_s the time decay constant for the membrane voltage and synaptic currents respectively.

Tempotron is a binary classifier, meaning it can only fire for a single class spike pattern and will remain silent for the other classes. Weight updates using Tempotron will only happen when an error occurs. If the neuron didn't fire for it's designated class, it synaptic weights will be **increased** by Δw_i. If the neuron fired for another class which isn't it designated class, the synaptic weights will **decrease** by Δw_i.

The synaptic weight update is defined as:

$$\Delta w_i = \lambda \sum_{t_i < t_{max}} K(t_{max} - t_i) \tag{3.12}$$

Here t_{max} is the time when the membrane voltage reaches it threshold voltage and λ is a constant which determines the maximum size of the weight update.

3.6.3 Unsupervised training: Spike Timing Dependent Plasticity & Triplet Spike Timing Dependent Plasticity

Spike Timing Dependent Plasticity (STDP)[27] is a unsupervised learning method which changes the synaptic weights based on the difference between the precise timing of pre-synaptic and postsynaptic spikes. The synaptic update rules are given by:

$$\Delta w_i = A_+ \exp\left(\frac{-\Delta t}{\tau_+}\right) \quad t_{pre} < t_{post} \tag{3.13}$$

$$\Delta w_i = -A_- \exp\left(\frac{-\Delta t}{\tau_-}\right) \quad t_{pre} > t_{post} \tag{3.14}$$

Here Δt is the difference between the pre-synaptic and postsynaptic spike, τ_+ and τ_- are the STDP learning windows and A_+ and A_- are the maximum potentiation and depression amplitudes. The synaptic weight will thus be increased when a pre-synaptic spike arrives before the postsynaptic spike and will decrease when it arrives after the postsynaptic spike.

There has been enormous amount of interest in STDP, due to it's relation to the biological neuron as found in [28]. However STDP suffers from instabilities when the two or more spikes arrive within the same learning window. This results in potentiation an depression in the same learning window. Triplet Spike Timing Dependent Plasticity (TSTDP) [29] can help alleviate this issue by adding an extra spike pair. The spike pairs used in TSTDP are illustrated in Figure 3.12.

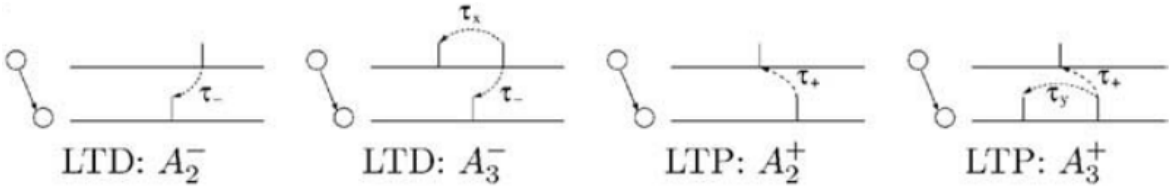


Figure 3.12: The long term depression(LTD) and long term potentiation(LTP) pairs [29]

This extra spike pair changes the synaptic update rules shown in eqs. (3.15) and (3.16).

$$\Delta w_i = A_{2^+} \exp\left(\frac{-\Delta t_1}{\tau_+}\right) + A_{3+} \exp\left(\frac{-\Delta t_2}{\tau_y}\right) \quad t_{pre} < t_{post} \tag{3.15}$$

$$\Delta w_i = -A_{2^-} \exp\left(\frac{-\Delta t_1}{\tau_-}\right) + A_{3^-} \exp\left(\frac{-\Delta t_3}{\tau_x}\right) \quad t_{pre} > t_{post} \tag{3.16}$$

STDP and TSTDP both have a very large design space with a lot of parameters, four for STDP and TSTDP has eight parameters this makes it quite difficult to find the right set of parameters to make sure training goes correct.

Another issue with STDP and TSTDP is the inherently instability of the methods. Synapse with high weights are more likely to spike and thus more likely to be strengthen than synapses with small weights, causing them to keep increasing until they reach their limit.

3.7 Self Organizing Map

The Self Organizing Map (SOM) [30, 31] invented by Theo Kohonen is an unsupervised ANN architecture capable of creating a low dimensional spatial representation of higher dimensional data in the form of spatial clusters. The SOM consists of a grid of nodes(seeFigure 3.13) and a input layer. The grid can have different topologies e.g. square or hexagonal. Each node in the grid contains a n-dimensional euclidean model vector m_j. Where j denotes the spatial index of the grid node associated with that vector

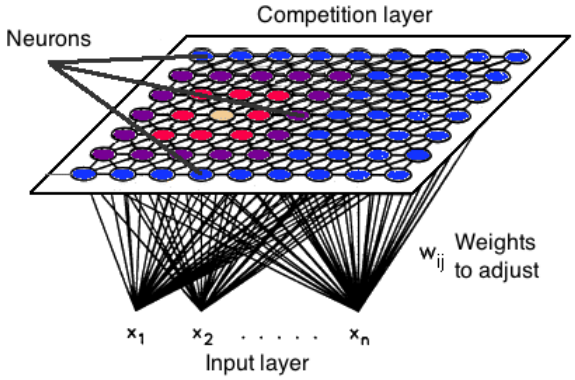


Figure 3.13: The Self Organizing Map [32]

Instead of using backpropagation for training, the SOM employs a competitive learning method. During each training step a vector $\mathbf{x}$ is presented at the input layer. The node whose model vector has the smallest euclidean distance to $\mathbf{x}$ is the "winner" or "Best Matching Unit"(BMU), denoted as c:

$$\mathrm{c} = \underset{j}{\operatorname{argmin}}\{\|\mathbf{x} - \mathbf{m}_j\|\} \tag{3.17}$$

with $\mathbf{x}$, $\mathbf{m}_j \in \mathbb{R}^n$. After finding the BMU, it's model vector and that of the local neighbours need to be updated to better match the input data. By also updating the neighbouring nodes, spatial clusters will be created within the grid. The model vector update is given by:

$$\mathbf{m}_j(\mathbf{t}+\mathbf{1}) = \mathbf{m}_j(\mathbf{t}) + h_{cj}(t)[\mathbf{x} - \mathbf{m}_j(\mathbf{t})] \tag{3.18}$$

with $h_{cj}(t)$ and t, the neighbourhood function centering around the BMU and the current training step respectively. A common neighbourhood function to use is:

$$h_{cj}(t) = \alpha(t) \exp\left(\frac{-\|c-j\|}{\sigma^2(t)}\right) \tag{3.19}$$

with $\alpha(t)$ the learning rate, j the index of a node in the grid, $\sigma(t)$ a scalar function determining the fall-off of $h_{cj}(t)$. The neighbourhood function dictates which nodes in the grid will receive an update to its model vector, bringing it closer to the BMU. Note

that $\alpha(t)$ and $\sigma(t)$ are dependent on the training steps. These monotonically decreasing functions, shrinking the neighbourhood function as training progresses.

After training unseen data can be presented to the SOM, creating clusters of data samples as shown in Figure 3.14. A SOM effectively maps a n-dimensional input vector to an output space $\mathbf{y}\ \epsilon\ \mathbb{R}^{MN}$, with M, N the length and width of the SOM. When $MN < n$, a dimensionality reduction on the input data has been applied.

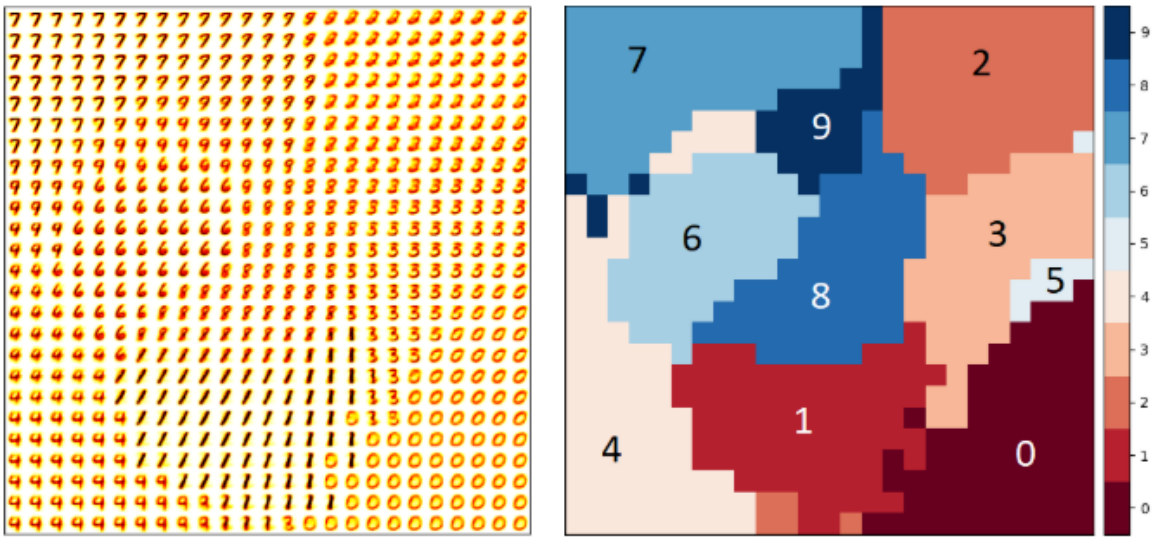


Figure 3.14: Clustering example: MNIST images in a 25x25 grid [33]

3.8 Spiking Self Organizing Map

The Spiking Self Organizing Map (SSOM) is a SNN structure, that tries to emulated the behaviour of the SOM using spikes. It has been a topic of various works [34, 35, 33, 24]. A common SSOM topology is illustrated in Figure 3.15. A grid of neurons, where each neuron is fully connected to the input layer and all-to-all connected within the grid. Due to a mix of inhibitory and excitatory connections between neurons, competition within the grid occurs.

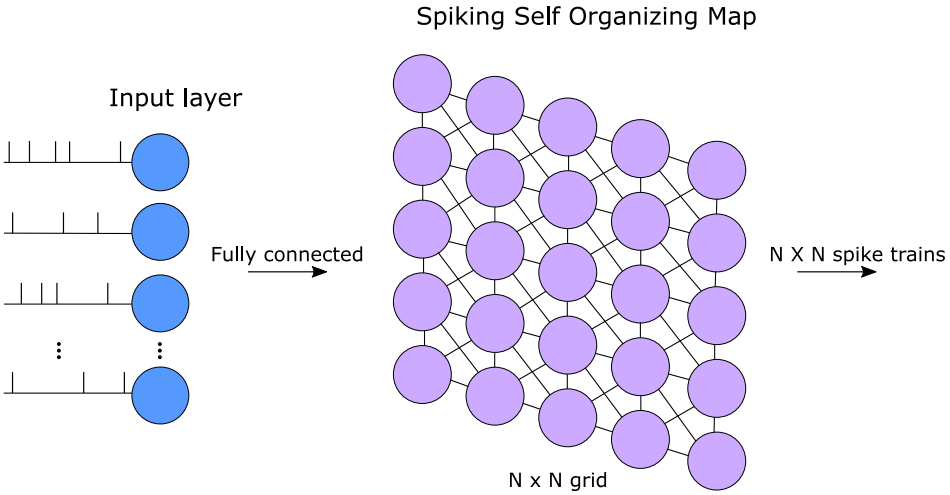


Figure 3.15: A spiking self organizing map structure

The assignment of the wining node in a conventional SOM, where the distance is calculated between the node's model vector and the input vector, has been replaced by learning on spike timings. When the SSOM has been trained, often by a unsupervised learning method, it can map spatial spike trains to lower dimensional spike train similar to the dimensionality reduction in the conventional SOM. The general behaviour of the SSOM is as follows. First, a spike pattern is presented at the input layer. These spikes will contribute to the neurons membrane voltage according synaptic weights, which were previously trained. Second, the neuron that has accumulated enough potential will elicit a spike, because it trained to fire for the incoming combinations of spikes. Lastly, the spiking neuron will excite any neurons within its close neighbourhood and inhibit the neurons which are spatially further away. This allows for certain groups of neurons to fire together based on specific input spike patterns.

3.9 Radar & The Doppler effect

Radar is a sensing system used for measuring the distance, velocity and location of objects, illustrated in Figure 3.16. An electromagnetic wave (EM) is transmitted via an antenna towards a target. This EM-wave will be partly reflected back to the antenna, entering the receiver.

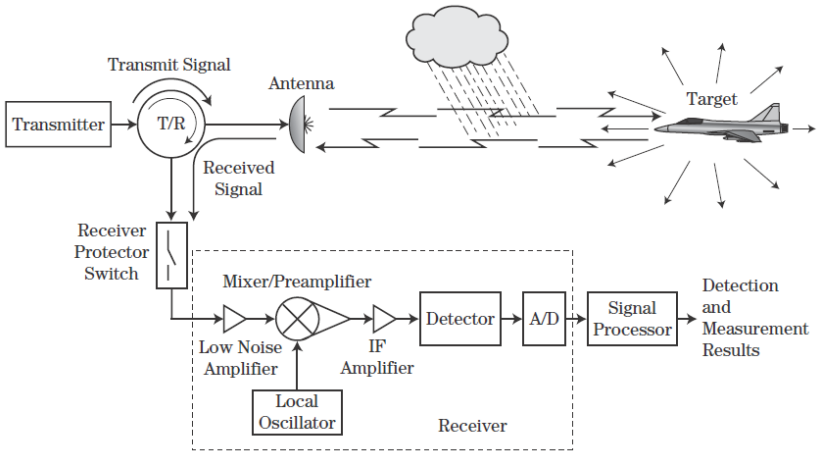


Figure 3.16: A basic radar system [36]

If the target moves with a certain velocity, the frequency of the EM-wave that is reflected back will have changed slightly due to the *Doppler effect*. This phenomena occurs when the source of a wave moves relative to the observer. This effect can be seen in any type of waves be it water, sound or electromagnetic. The observed frequency can be described as:

$$f = \left(\frac{c \pm v_r}{c \pm v_s} \right) f_0 \tag{3.20}$$

with c, v_r, v_s f_0, the wave propagation speed in the medium (e.g speed of light), velocity of the receiver, velocity of the source and the transmitted frequency respectively. By measuring f, eq. (3.20) can be solved for the velocity of the target v_s.

The range of the target can be found by measuring the time-of-flight(TOF):

$$R = \frac{ct}{2} \tag{3.21}$$

with R the range to the target and t the TOF time. Note that we need to divide by two since wave travels to the target and back.

3.10 Micro-Doppler effect

As discussed in the previous section, the Doppler effect in radar occurs when a target moves through a medium and reflects back an EM-wave. However, an object often vibrate and rotates when moving, causing additional frequency modulation on the returning signal will appear as extra frequencies around the signal's Doppler frequency, known as the Micro-Doppler effect. First describe by Chen et al [37, 38] for radar, based on [39] Chen describes in detail the effect of rotation, vibration, tumbling(rotation, translation and acceleration) and coning(rotation around axis) on the Doppler frequency.

This effect is illustrated in Figure 3.17 . It shows a Micro-Doppler signature of a simulated helicopter with four rotor blades. The four blades rotate with a fixed period, showing sinusoidal behaviour around the Doppler shifted observed frequency at ∼-2 kHz. These Micro-Doppler signatures reveal additional information about the dynamics of the target instead of just range and velocity information.

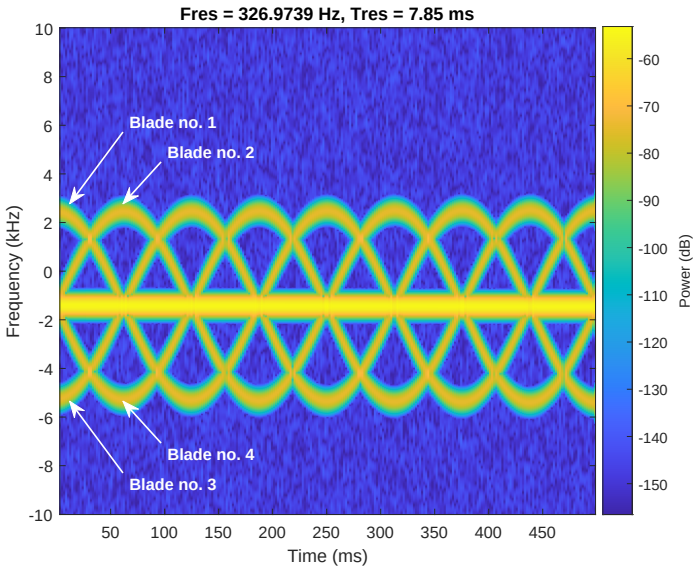


Figure 3.17: The Micro-Doppler signature of a simulated helicopter with four rotor blades [40]

4 Methods

4.1 Radar dataset

Radar data can be structured into several formats: time-amplitude, range-amplitude, time-range, time-Doppler, range-Doppler and time-frequency [41]. This leaves a broad range of the data format to choose from. However since recording and creating a radar dataset is out of the scope of this thesis project and thus a existing dataset is needed. Unfortunately few public available datasets exist. Two datasets were found as possible candidates. First, a dataset recorded with Google's Soli [3] radar sensor by [9] containing 2D range-Doppler frames over time. Second, the DopNet dataset [4, 42] containing Micro-Doppler spectrograms in the time-Doppler format.

It was decided to use the DopNet dataset instead of the Soli dataset. Mainly due to format of the data, since the Soli dataset has dimensionality *Doppler* x *range* x *time*, while DopNet has the dimensionality *Doppler* x *time*, dropping the range information. This reduction in dimensionality is useful when moving to a hardware constraint design where computing resources are limited. Since we're not using any range information, gestures that move away or towards the sensor will be hard to recognize.

DopNet contains four different hand gestures recorded from six individuals. The gestures classes in the dataset are illustrated in Figure 4.1.

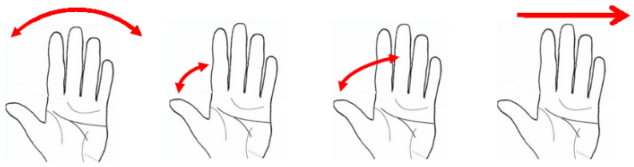


Figure 4.1: The handgestures in DopNet from l.t.r Wave, Click, Pinch & Swipe [42]

The Micro-Doppler signatures for each of the hand gestures are shown in fig. 4.2. Each hand gesture has an distinct signature, however there are similarities between the gestures. For example the Pinch and Swipe gesture, both spatial flat. The time duration of each performed gesture also differs between classes, a wave gesture takes a significant large time to perform then e.g. the click gesture. The spectrograms have a frequency range of -400 Hz to 400 Hz around the observed frequency.

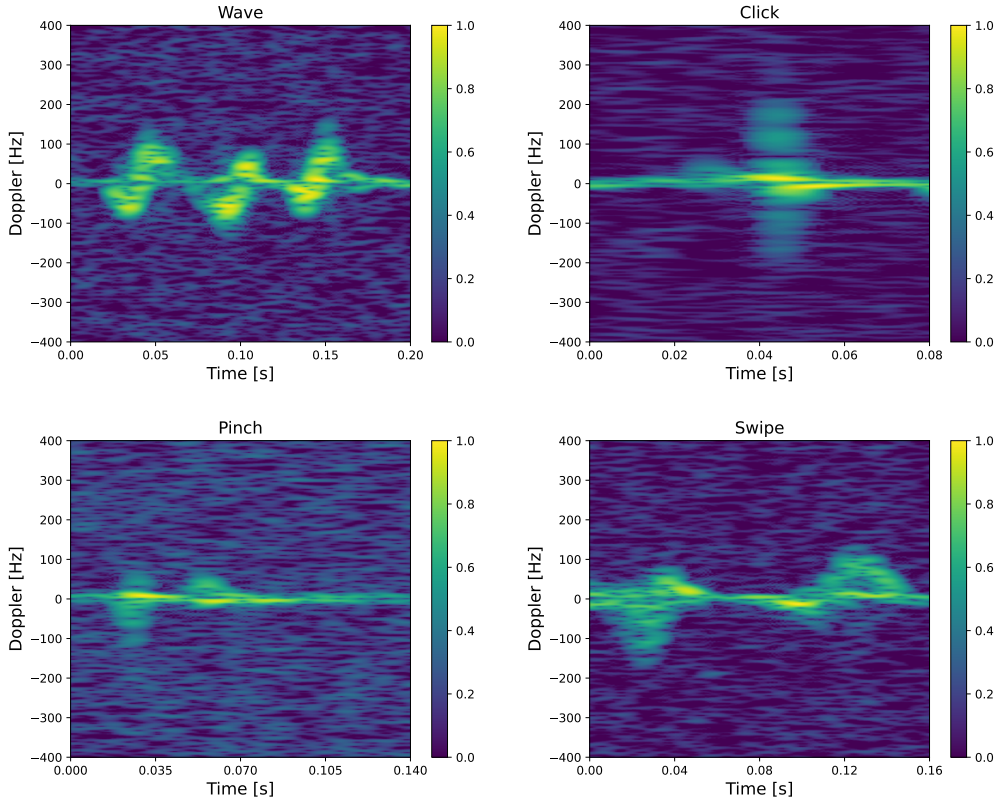


Figure 4.2: The Micro-Doppler signatures of the four gestures; Wave, Click, Pinch and Swipe

DopNet already comes partly pre-processed, the steps performed are described in [4]. The final preprocessing step done here is feature scaling (min-max normalization) to normalize the value to [0,1] :

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4.1}$$

4.2 SNN classifier

Since our aim is to design a architecture intended for low power applications we'll limit ourself to a temporal encoding scheme. Simply because rate-based encoding uses significantly more power than a temporal-based scheme. This has a major influence on the decision for the learning method and effectively the resulting network topology. Methods using a rate-based approach e.g. ANN-SNN conversion are not desired.

The training method that will be used for the SNN classifier is the Tempotron learning method(see Section 3.6). Tempotron is a simple training method with only 3 parameters to set. This makes the design space small and workable.

Using Tempotron limits us to single layer networks due to the nature of this learning rule. Since DopNet contains 4 different classes the classifier will be a single layer SNN with 4 neurons for each class one neuron.

The complete pipeline of the proposed SNN classifier is illustrated in Figure 4.3

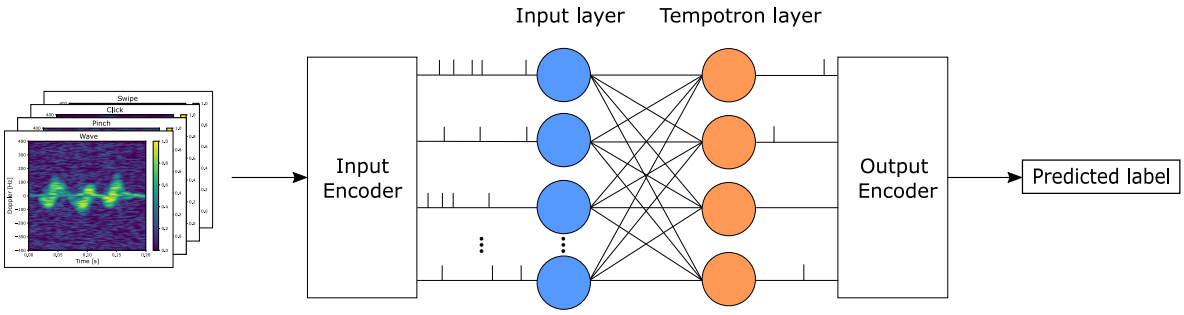


Figure 4.3: The SNN classifier

The universal approximation theorem states that ANNs can approximate a wide range of functions. Often complex problems require multiple hidden layers to achieve proper approximation of the desired function. This also holds for SNN architectures and the lack of gradient based learning in SNN is often the reason why SNNs generally lack behind in terms of accuracy.

Since we cannot make the approximation more complex, we need to make the problem to be solved simpler. One way of doing this is by reducing the dimensionality of the input data, also called dimensionality reduction or feature reduction. Dimensionality reduction for ANNs have been a popular research topic for many years and numerous methods exists e.g. principal component analysis (PCA) & UMAP. As the goals is to implement a complete SNN architecture mappable to hardware, a dimensionality reduction using spiking neurons is desired. A proposed solution will be discussed in the next section.

4.3 Proposed architecture

In Section 3.7 we discussed the network known as the Self Organizing Map, an unsupervised clustering method capable of transforming an input sample to a lower spatial dimension. A spiking variant exists as we saw in Section 3.8. The Spiking Self Organizing Map (SSOM) is a good candidate for the feature reduction we're seeking, as it uses spikes. As long as the amount of neurons in the grid does not exceed any hardware constraints and the hardware platform supports a grid structure with all-to-all connections within the layer, it can be mapped to hardware.

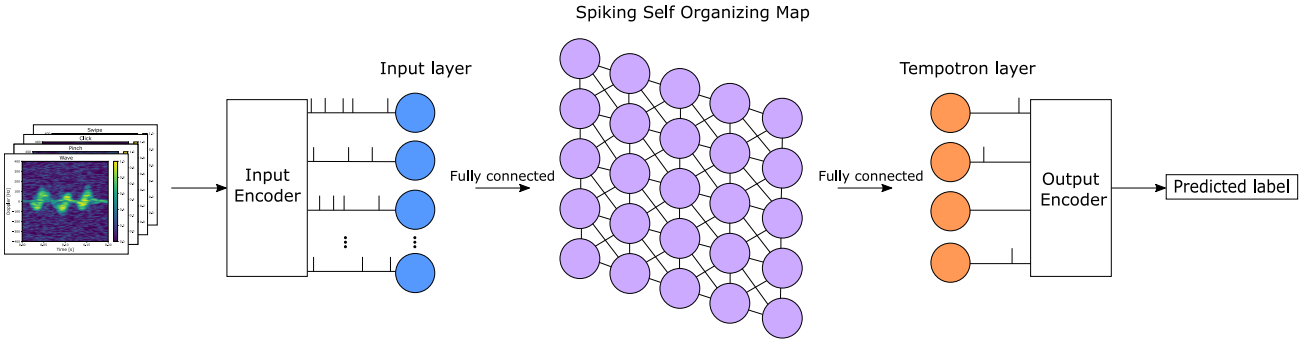


Figure 4.4: The complete architecture with SSOM extention

By extending the SNN classifier with the SSOM, the complete pipeline becomes as shown in Figure 4.4. As the spectrogram enters the network, it will go into the input encoder to be encoded into spike trains. After encoding the spike trains will be presented to the SSOM where they are transformed to the lower dimensional space. The final layer trained by Tempotron will receive the transformed spike trains and try to perform the classification. At the last stage of the pipeline the decoder will interpret the final layer's output spike as a prediction.

The implementation of the SSOM is based on [24] as it was already part of the SNN simulator used for the experiments. The synaptic weights between the input encoder and the SSOM are trained using the Spike Timing Dependent algorithm (see Section 3.6) to emulate the unsupervised nature seen in a normal SOM.

The synaptic weights in the grid are scaled using a neighbourhood function called the "Mexican hat" function. This is to ensure competition between the grid neurons, as one neuron fires it needs to inhibit the other neurons preventing them from firing. The neighbourhood function is defined as:

$$h(d) = (1+\alpha)\exp\left(\frac{-d^2}{2r^2}\right) - \alpha\exp\left(\frac{-d^2}{2(\beta r)^2}\right) \tag{4.2}$$

with α, β, r, d, the inhibitory magnitude, the inhibitory spread, radius of the neighbourhood and distance to other neurons receptively.

Note that the radius of the neighbourhood function does not change over time, which is different from the neighbourhood function of the SOM, see Equation (3.19). This difference could influence the way the SSOM creates it's clusters, as it's implemented in the SOM to ensure that during the final training steps only the winning neuron gets updated. It was possible to make the neighbourhood decrease over time, however this required alterations to be made on the core codebase of the SNN simulator and this was deemed out of scope for the project.

4.4 Encoder

As we know from Section 3.5, rate based encoders elicit a lot more spikes w.r.t temporal encoders. This makes rate encoder more power consuming as they need more spikes to represent a certain value. However, temporal encoders are more susceptible to noisy data as a single spike holds more value in sparse precise spike trains than in rate encoded spike trains. Energy consumption needs to be minimal when designing for hardware, so a temporal-based encoder is required.

For the encoder, the population threshold encoding (PTE) algorithm is used. Pan et al [43] have shown the effectiveness of PTE on spatial temporal audio data. The algorithm employs two neuron populations, called *onset* and *offset* neurons. Each one of these neurons have their own firing threshold from a set of equally distributed thresholds, illustrated in Figure 4.5. Here the onset neurons are given in rend, the offset neurons in blue and the dashed lines denote the threshold levels.

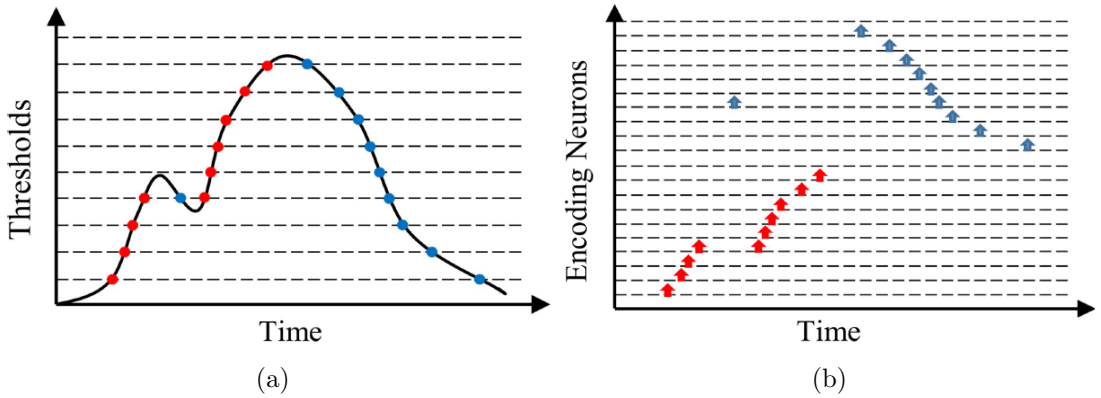


Figure 4.5: The Population Threshold encoding (a) a signal encoded with 10 thresholds (b) the resulting spike trains with 20 encoding neurons.[43]

When the signal crosses a threshold, the neuron related to this threshold will spike. If the signal crosses a threshold from a lower to a higher value, a neuron in the onset population (red) will elicit a spike, when the signal crosses a threshold from a higher value to a lower value that will elicit a spike in the offset population (blue). The pseudo code for the PTE algorithm is given in Algorithm 1. Figure 4.6 illustrates the encoding of two spectrograms of gestures wave and swipe into spike trains, with five thresholds. The spike trains conserve the spatial features shown in the spectrograms. Note that the spectrograms shown here don't range from 400 Hz to -400 Hz, instead ranging from 100 Hz to -100 Hz. This is also visible from the amount of encoding neurons, which is 2000 since 200 frequencies x 5 thresholds x 2 neurons per threshold(onset & offset).

The PTE algorithm is a simple encoding scheme, but comes at the cost of increasingly hardware cost for higher dimensional data as the amount of encoding neurons increase with $2N$ for each signal value you want to encode, with N the number of thresholds. In [43] audio spectrograms with only 20 frequency bins, while DopNet has 800 frequency bins. This creates a trade off between the amount of resources and the spatial resolution created by the number of thresholds, which will be discussed further in Section 5.4.

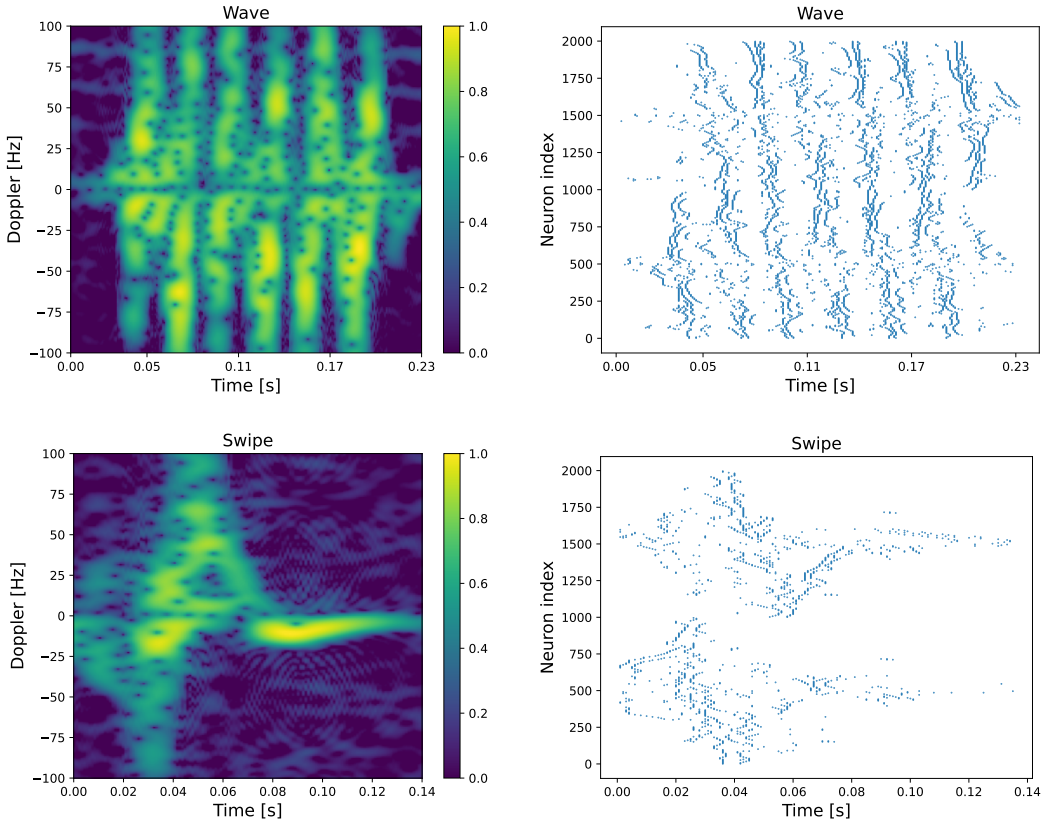


Figure 4.6: The gesture wave and swipe encoded with PTE using 5 thresholds. Here a subset of the data was taken from 100 Hz to-100 Hz

Algorithm 1: Population Threshold encoding (PTE)

Data: S, a 1xN vector containing the signal values
T, a 1xM vector containing the thresholds
Result: *onset_spikes*, a NxM matrix containing onset neuron spikes
offset_spikes, a NxM matrix containing onset neuron spikes
onset_spikes $\leftarrow zeros(N, M)$
offset_spikes $\leftarrow zeros(N, M)$
foreach $n \epsilon N$ **do**
  foreach $m \epsilon M$ **do**
    if *S(n-1)* $<$ *T(m)* ***AND*** *S(n)* $>$ *T(m)* **then**
      onset_spikes(n,m) = 1;
    end
    if *S(n-1)* $>$ *T(m)* ***AND*** *S(n)* $<$ *T(m)* **then**
      offset_spikes(n,m) = 1;
    end
  end
end

4.5 Decoder

Due to the nature of the Tempotron learning method, when we present a gesture to our architecture only one neuron in the output layer should fire. This makes decoding the output of the final layer very simple, since each neuron is associated with a class label. The Time-to-first-spike decoder, shown Figure 4.7 would be a good fit for our decoder. It will make a prediction on the class who's neuron has fired first. In Figure 4.7 there are four neurons, neuron two fired first, therefore it predicts that the spike train belongs to class two. Note that the decoder assumes a correlation between the spatial location of the neuron and the classes.

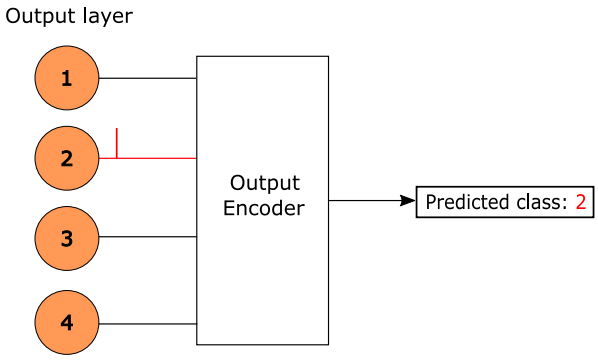


Figure 4.7: An output layer with four neurons and a First-time-to-spike decoder

Since this decoder only needs one spike to make a prediction, makes it very susceptible to noise. A single noisy spike can trigger a wrong prediction. However we assume that in combination with a layer trained by Tempotron, this will not major impact on the overall classification accuracy.

4.6 Baseline architectures

Little literature [42, 4] was available on the classification performance using DopNet during the early stages of the project. At the early stages the known state of the art was as given in Table 4.1.

Table 4.1: State of the art accuracies for DopNet at the early stages of the project

Algorithm	Accuracy	Reference
SVM Quadratic	74.2 %	Ritchie et al (2019) [42]
K Nearest Neighbor	87.0 %	Ritchie et al (2020) [4]

All these methods are machine learning techniques and don't use any ANNs. It was not guaranteed that a SNN based solution would achieve the same levels of accuracy, mainly because machine learning methods approach the problem in a whole different way. Therefore, it was decided that a baseline network was needed using ANNs. Because it would give a better estimation of the maximum achievable accuracy with a SNN, since ANNs an SNNs are more relatable then any machine learning approach.

There are multiple ways to classify the DopNet dataset with ANNs. One approach is to view the spectrograms as images and ignore any temporal information. Then classify them using Convolutional Neural Networks (CNNs), large networks primarily used for image recognition. A second approach is to use Recurrent Neural Networks (RNNs), used to classify or predict time series data. These network use memory elements and recurrent connections to learn temporal relation in the data.

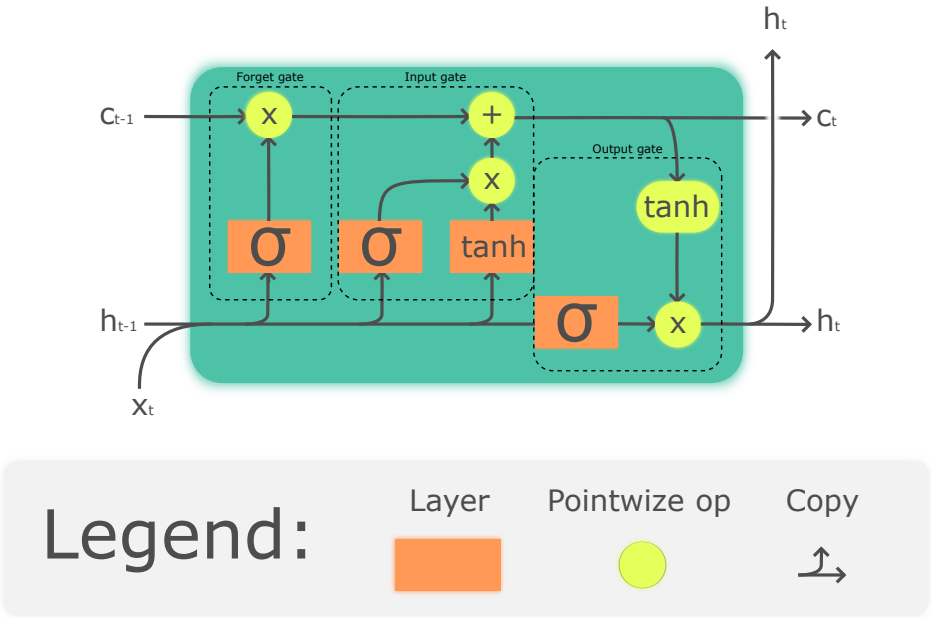


Figure 4.8: A LSTM cell [45]

One of the main benefits of a SNN are its inherent temporal dynamics. Thus ignoring any temporal information might give the wrong impression on actual classification performance. Using a RNN is therefore desirable. Long Short Term Memory (LSTM) [44] is a popular RNN architecture, used for the prediction of time series data and natural language processing. They consists of computational nodes called cells, illustrated in Figure 4.8. Each cell has a memory, called the cell state C_t for holding any relevant information. This cell state is altered by the a series of gates. First the forget gate decides to what extent we need to delete information from the previous time step. Second, the input gate determines what information is written to C_t based on the cells input x_t and the hidden state of the previous cell h_{t-1}. Finally the output gate generates the output h_t based on the cell's current state, x_t and h_{t-1}. The cell's current state c_t and hidden state h_t are passed to the next cell for processing.

The LSTM classifier consists of series of cascaded LSTM cell, who's output is connected to a fully connected output layer. The LSTM classifier which will used to classify the DopNet dataset is illustrated in Figure 4.9. The output layer consists of four neurons, since there are four classes, with activation function softmax.

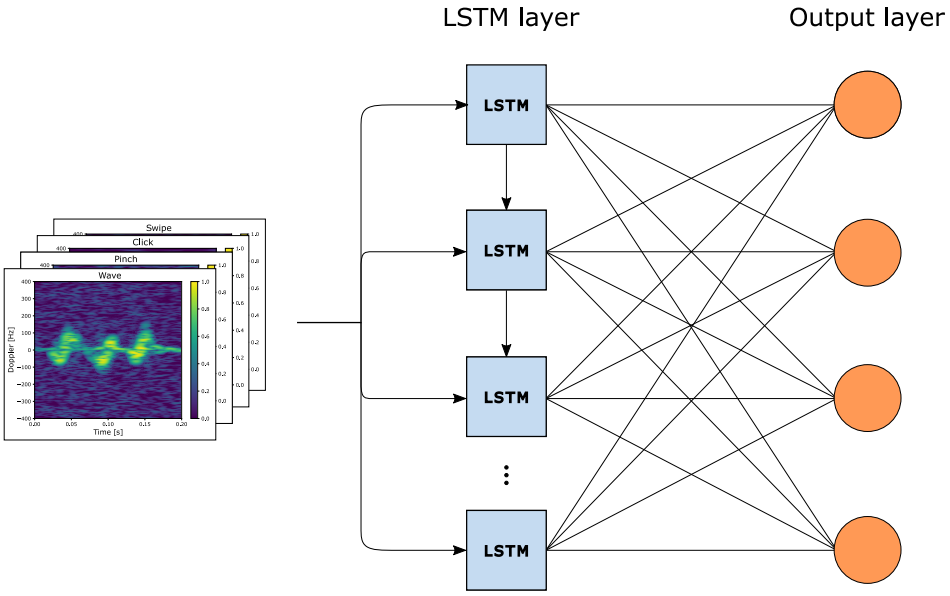


Figure 4.9: The baseline classifier using Long Term Short Memory cells

SOM-LSTM architecture

Little was known about the effects of mapping the DopNet dataset to a lower dimensional space using a SOM or spiking SOM, literature was effectively non-existent. Therefore a Self Organizing Map was implemented to test the expected behaviour early on as a means to verify any assumptions before moving on to a spiking version of the SOM. By extending the LSTM classifier with a SOM, we can test if the LSTM benefits from a smaller data dimensionality and in turn make an estimation if this would also be the case when used in combination with our SNN classifier. The extended architecture is illustrated in Figure 4.10. Note that between the SOM and LSTM cells a new stage has been added, the Embedding matrix. A concept often used in natural language processing (NLP), maps the output of the SOM which are locations of the neuron to vectors of real values.

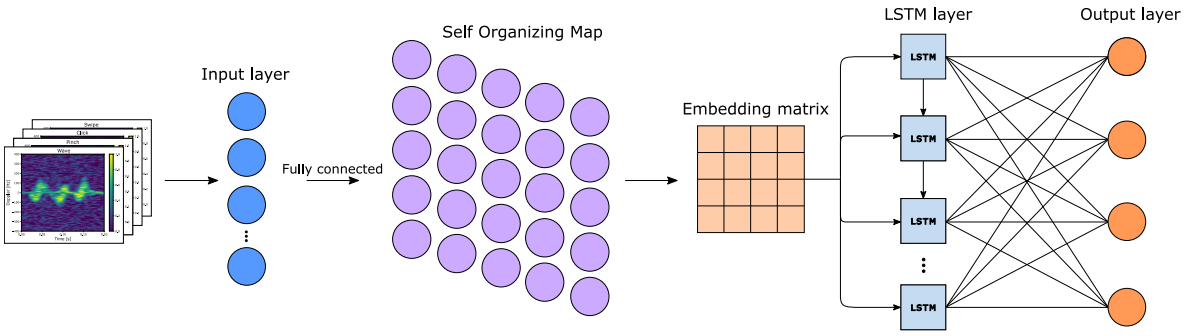


Figure 4.10: The extension of the baseline classifier with a Self Organizing Map

5 Experiments & Results

5.1 Spiking Neural Network Simulator

The lack of a neuromorphic hardware platform to test the proposed designs, limits us to a simulation environment. Various open source simulating environments already exist and actively maintained [46, 47, 48, 49].

It's important to note that the simulations done in this work try to approximate the behaviour of the neuromorphic hardware as close as possible. However, the design cannot be directly mapped to hardware. Since moving from a software simulation to a design running on hardware requires a lot of extra steps, deemed out of scope for this project.

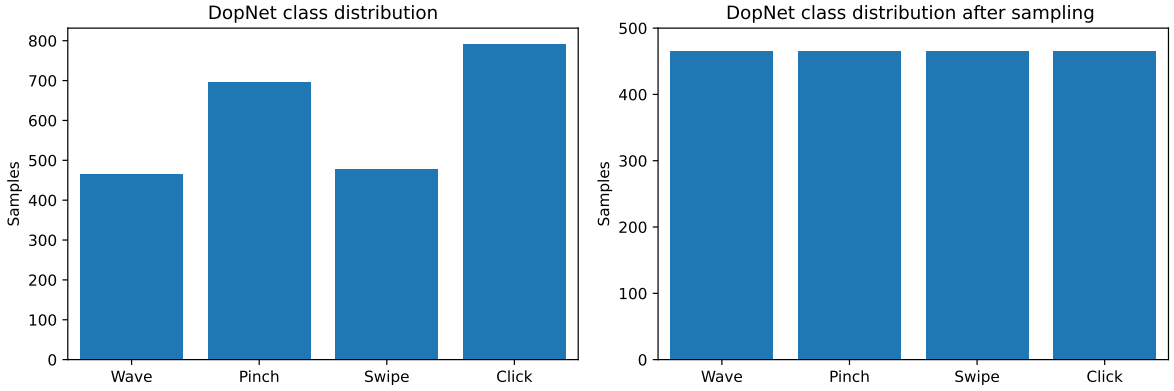
5.2 Experiment setup

Balanced dataset

DopNet is an imbalanced dataset as shown in Figure 5.1a. Using the dataset like this to train and validate the models, might increase the chance of overfitting on the *Click* or *Pinch* class. To make the dataset balanced, either samples need to be delete (sampling) or the dataset can be expanded with the use of data augmentation. It was decided to apply data deletion instead of expanding the dataset with augmented samples. Mainly due to the unknown effects of these augmented samples on the ability to generalize.

Also augmenting radar spectrogram data is not a simple task. According to [50], "traditional augmentation techniques, like translating and resizing cannot be applied for radar data". In our case, the Doppler information would be transformed in such a way it's no longer representable of the original data.

Random sampling is applied on DopNet, while enforcing an equal distribution across the classes. Resulting in the class distribution shown in Figure 5.1b.



(a) (b)

Figure 5.1: The class distribution in the DopNet dataset, (a) original (b) after random sampling

Stratified k-fold cross-validation

To validate how the proposed models generalize on unseen data an independent test data set is needed. Therefore, we'll be using the validation technique called stratified K-fold cross-validation. It splits the entire dataset into k folds of equal size. One of these folds will be selected as the independent test set and the remaining folds will be used as training data, then the model is trained and validated. This process is repeated k times until each of the folds has been used as the test set at least once, this is illustrated in fig. 5.2 for 5 folds. Stratified cross validation also makes sure that the class distributions inside the training set and test set are equal in all the folds, in order to prevent class imbalances. K-fold cross validation is often employed to optimize a model's hyperparameters across all the folds. In this work cross validation is only used to produce performance metrics, no optimization of hyperparameters is done here.

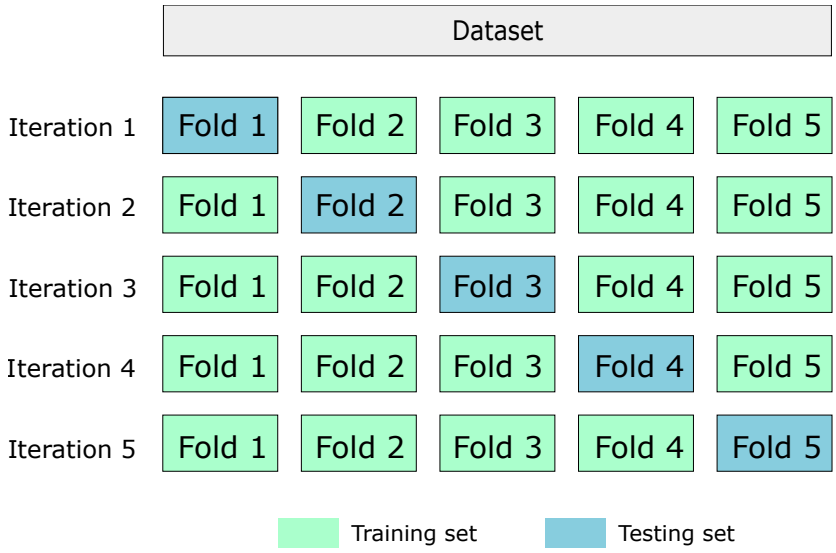


Figure 5.2: The separation of a dataset using 5-fold cross validation

5.3 Baseline architectures experiments

LSTM classifier

As discussed in Section 4.6, a LSTM classifier was implemented to get a initial impression of the capabilities of the DopNet dataset. Tensorflow[1] a deep learning python toolbox with GPU support is used to implement, train and test this classifier. The parameters for this classifier are presented in Table 5.1.

Table 5.1: Parameters for the LSTM classifier

Parameter	Value
Output layer size	4 neurons
Epochs	50
Activation function LSTM	*tanh*
Activation function output layer	*softmax*

Cross validation was done for varying number of LSTM cells, illustrated in Figure 5.3. Smaller LSTM layers tend to perform worse as they has there's a smaller collective memory present in the network. This particular clear for a LSTM layer with a single LSTM cell, it cannot train on the hidden state from previous cells, only on the input data. With two cells, the accuracy shows significant improvement to $\backsim$ 60 % , this is expected as now the second cell can make use of the information in the hidden state of the first cell. However as the mean classification accuracy has increased, the standard deviation is still quite significant. Showing that two cells are not enough to make a model which can generalize enough on the data. As the LSTM layer gets larger the mean accuracy converges $\backsim$ 73 %, with an average standard deviation $\backsim$ 3%. Note that after an LSTM layer size of 12, adding additional cells doesn't lead to any significant gain in performance.

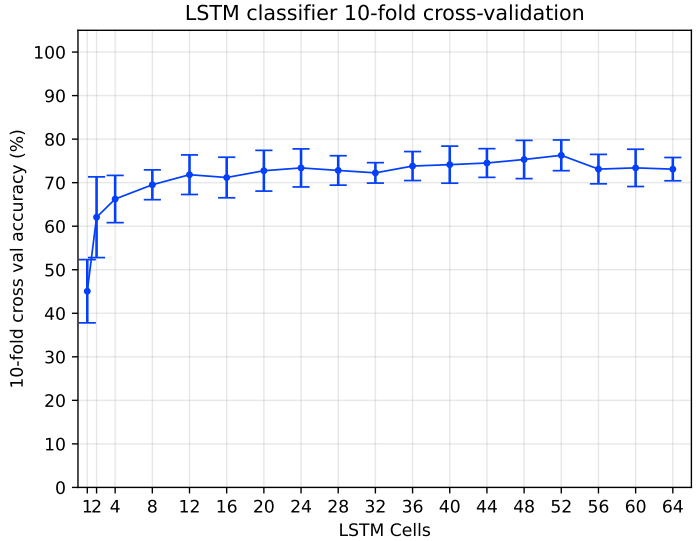
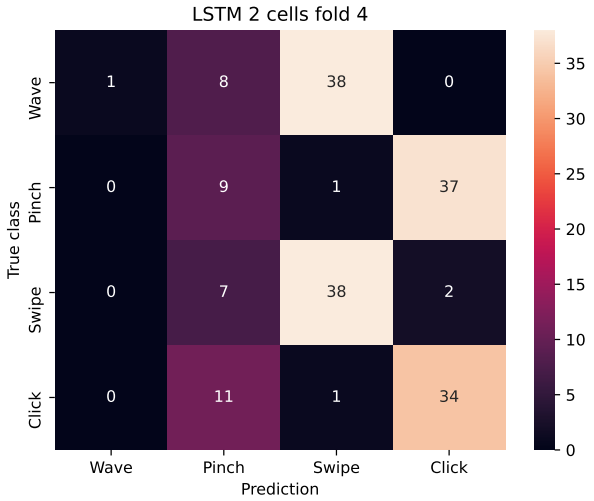
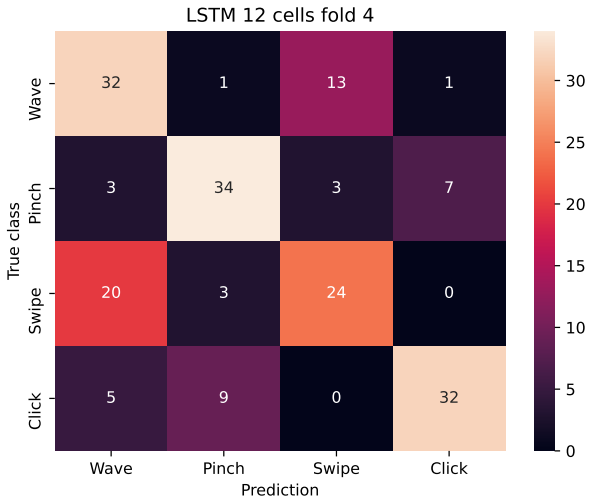


Figure 5.3: 10-fold stratified cross validation classification accuracy for varying LSTM sizes. For an numerical overview of the results see Table A.1.

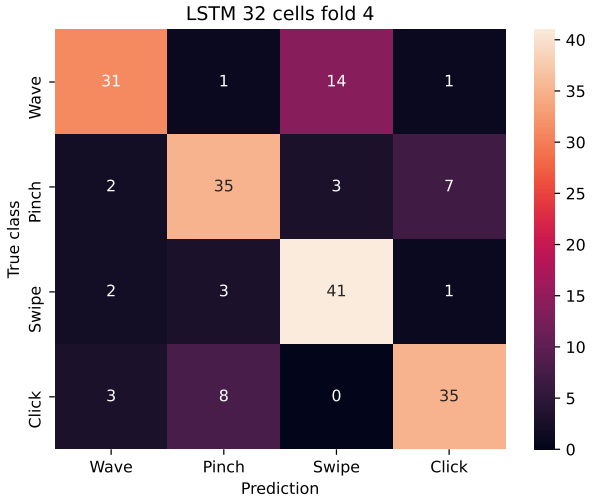
With multi-class classification a single accuracy metric can give the wrong impression. As overfitting may cause the overall accuracy to be high, other classes might be misclassified. Figure 5.4 shows the confusion matrices of 3 different configuration (2, 12 & 32 LSTM cells) using fold 4 as the testing set. Overfitting on the Swipe and Click class when using 2 cells can be observed. Interestingly enough, a correlation exists between the Wave and Swipe class and the Pinch and click class respectively. This makes sense as these sets of gestures due contain similar motions and the model with 2 cells has problems distinguishing these correlations. This partly solved by adding more cells to the LSTM layer as can be seen from the other confusion matrices. However, the correlation is still apparent as the amount of samples of the class Wave getting miss classified as Swipe($\backsim$ 13 samples) remains constant. Same for the Pinch-Click combination, where a constant amount of $\backsim$ 8 samples are misclassified. This might be solved by making the LSTM layer even larger. The constant misclassification of these samples might be caused by the spectral signature being to similar to a sample of another class. This correlation is already an interesting find and important to remember when evaluating the results of the SNN architecture.



(a)



(b)



(c)

Figure 5.4: Confusion matrices using fold 4 as test set with LSTM sizes (a) 2 cells (b) 12 cells (c) 32 cells

SOM-LSTM architecture

We're interested to see if the addition of the Self Organizing Map to the SNN classifier has a positive effect on the classification accuracy thus if transforming the input features to a smaller feature space is useful. To get a rough estimate on this behaviour, we performed some experiments on the baseline LSTM classifier extended with a SOM.

The network shown in Figure 4.10 was tested with the parameters presented in Table 5.2. Here it was chosen to keep the size of the LSTM fixed to 32 cell. Which is an acceptable size, as it's not the goal to find the most optimized network in terms of size, but rather explore the effect of adding the SOM to the pipeline. By testing against various sizes of the SOM grid, we can see the influence of the size of the grid on the classification accuracy, illustrated in Figure 5.5.

Table 5.2: Parameters for the SOM-LSTM architecture

Parameter	Value
Output layer size	4 neurons
LSTM cells	32
Epochs LSTM	50
Activation function LSTM	*tanh*
Activation function output layer	*softmax*
Epochs SOM	50
Learning rate SOM	0.1

For smaller grid sizes the classifier seems to perform worse, a trend previously seen with the LSTM classifier. Here it's likely caused due to a too small dimensionality, spatial information is lost due compression on a too small output space. Not enough neurons are present to create spatially separated clusters. As the grid neurons keep increasing as does the classification accuracy and when using 100 grid neurons the accuracy is at the same level as the LSTM classifier. Strangely when using a 12x12 grid(144 neurons) a decline is observed. This was unexpected, as we would expect a increasing or stagnate trend and not declining based on the trends we've seen earlier. However when using a 14x14 grid (196 neurons) or larger we observe a gain of $\backsim$ 10 % compared to the LSTM classifier. This seems to be the most significant accuracy improvement with respect to the LSTM. Note that the standard deviation of the accuracy is quite high especially for the smaller grid size, it does decrease for the larger grids.

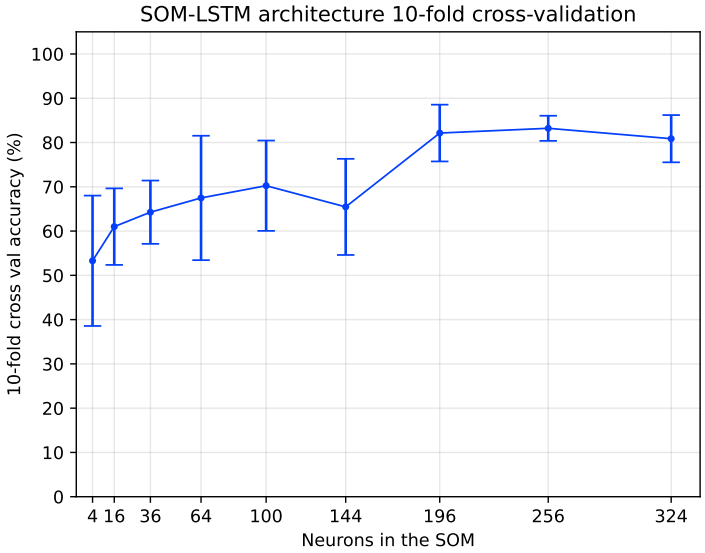
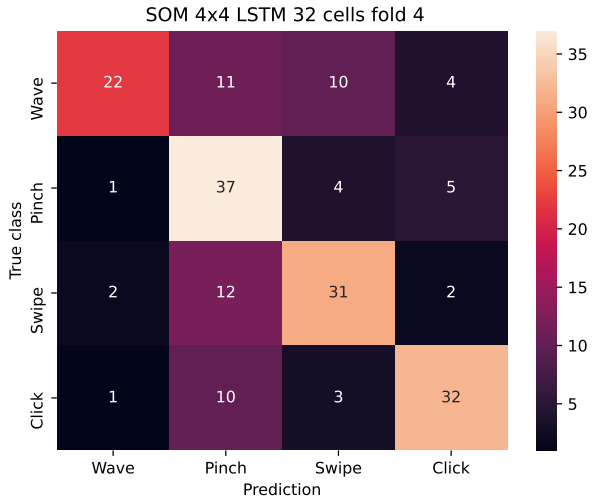
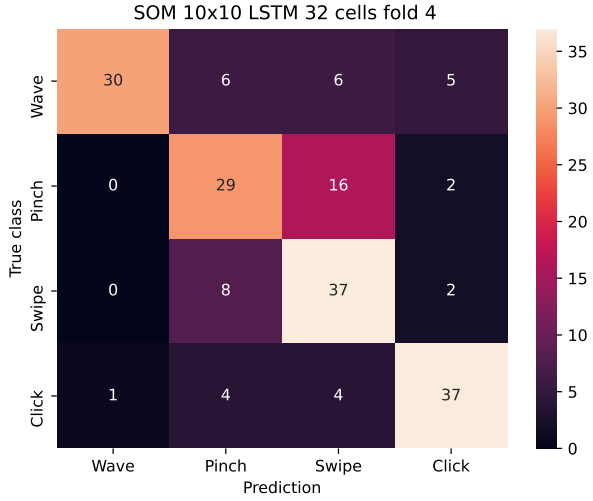


Figure 5.5: 10-fold stratified cross validation classification accuracy for varying amount of neurons in the SOM, with LSTM size 32. For an numerical overview of the results see Table A.2

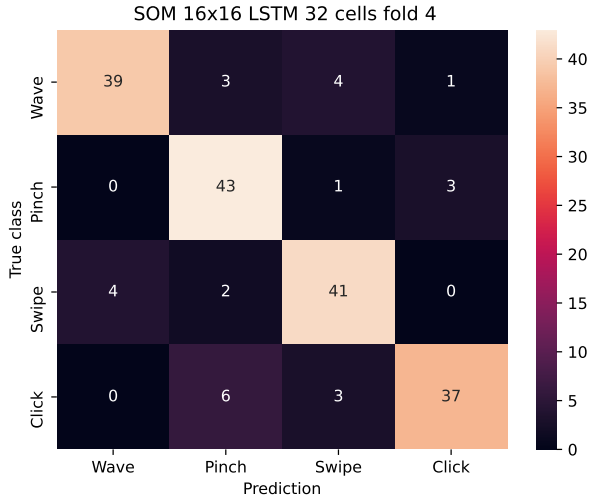
Once again let's investigate the confusion matrices for different grid sizes, shown in Figure 5.6. For the 4x4 grid a clear overfitting on the Pinch class is shown as major part of samples belonging to other classes are predicted as Pinch. Increasing the grid to a 10x10 seems to improve things and increasing the size of the grid even further achieves the best accuracy seen until now. The correlation between the two sets of classes we saw in Figure 5.4 is still present but the amount of classes that are getting misclassified has been reduced.



(a)



(b)



(c)

Figure 5.6: Confusion matrices using fold 4 as test set with SOM grid sizes (a) 4x4 grid (b) 10x10 grid (c) 16x16 grid

5.4 SNN classifier

The learning parameters in Table 5.3 are assumed for the SNN classifier. As the SNN simulator currently doesn't support any mechanism for early stopping, the Tempotron algorithm will run for the complete amount of epochs. During training only one output neuron will have the chance to spike, because Tempotron will interrupt the incoming spikes and perform a weight update at the first output spike. Therefore, t_{ref} has been set to 0 s as it does not contribute to the assumed behaviour in Tempotron.

Table 5.3: Parameters for the Tempotron learning method. The synaptic weights are initialized following a uniform distribution between -1 fC and 1 fC, here the uniform distribution is denoted as U. Also, τ_s is set close to zero as the simulator assumes spikes are modelled as instantaneous events

Parameter	Value
Initial synaptic weights	U(-1e-15, 1e15) C
Epochs	100
learning rate	5e-17
τ_m	0.1 ms
τ_s	0.33 μs
T_{ref}	0 s

We know that the number of thresholds in the encoder, partly determines the amount of neurons in the input layer. Adding more thresholds allows the encoder to be more sensitive to small changes in the signal at the cost of more neurons. Some encoder thresholds with their respective spacing ,the resulting input neurons and the 10 fold classification accuracy are presented in Table 5.4. Each additional threshold we add, results in an increase of 2 x 800 = 1600 input neurons. The design choice for the encoder thresholds has a big impact on the required hardware resources, we're interested in finding the relation between encoder thresholds and the classification accuracy. Smaller thresholds where not investigated further as the spacing was too large to capture any features.

A positive trend can be observed for the increasing thresholds. This is somewhat expected as adding more thresholds allows more information to be converted into the spikes trains. However, for 12 thresholds there appears to be a decrease in accuracy. This could be caused by an oversensitive encoder, where small changes in the signal caused by noise result in more noisy spikes.

Table 5.4: Amount of input neurons for thresholds

Encoder thresholds	Threshold spacing	Input neurons	Accuracy
4	0.250	6400	56.06 ± 3.23 %
6	0.167	9600	56.72 ± 3.25 %
8	0.125	12.8k	57.63 ± 3.47 %
10	0.100	16k	58.81 ± 2.60 %
12	0.083	19.2k	53.66 ± 7.54 %

From a hardware perspective the size of the required input layer is hard to realize especially when targeting an edge computing platform. Fortunately, *feature selection* can be applied to our data,which will reduce the size of the input layer. The Micro-Doppler signatures in Figure 4.2 show that the majority of the energy is confined to a specific frequency range. While a majority of the frequency bins are filled with background noise. This means a subset of the data can be selected without losing important features. Four different feature selection scenarios are explored , shown in Figure 5.7.

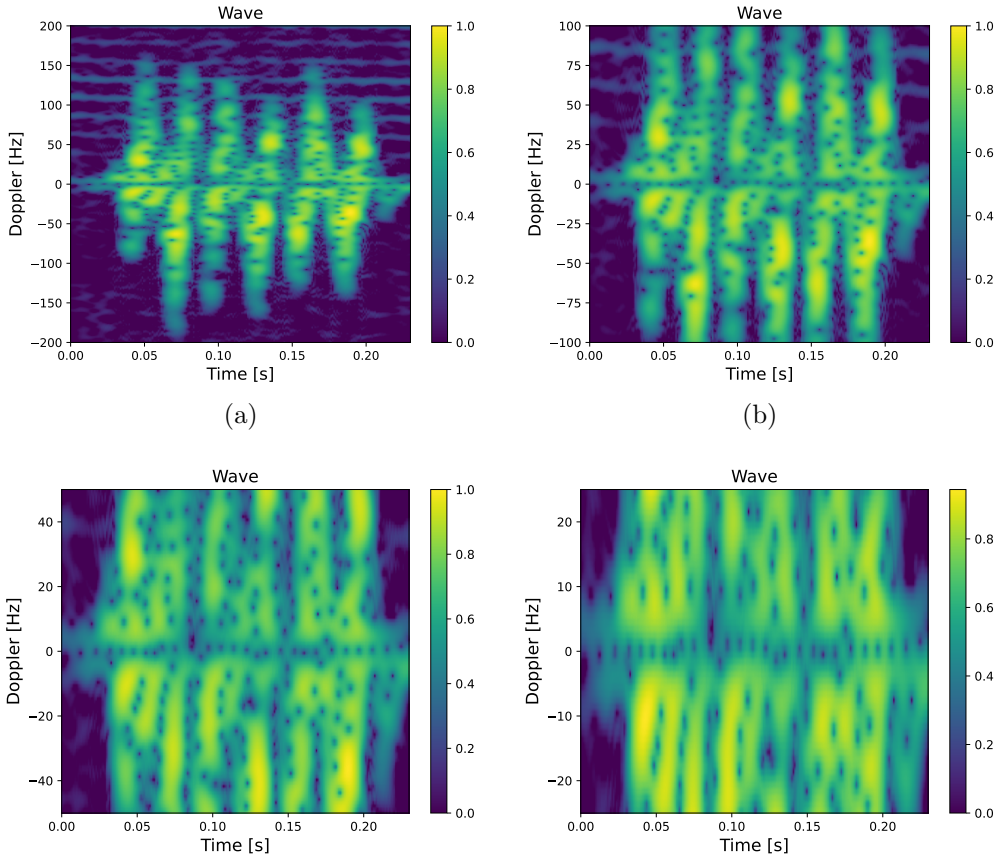


Figure 5.7: The four feature selection scenarios (a) -200 Hz to 200 Hz (b) -100 Hz to 100 Hz (c) -50 Hz to 50 Hz (d) -25 Hz to 25 Hz

We expect the accuracy to remain on the same level as in the previous experiments for the scenarios in Figure 5.7a & 5.7b, but the smaller selections (Figure 5.7a & 5.7b) are likely to show a decrease in performance as spatial features are being discarded. The previous experiments are repeated for these four different scenarios, giving the results presented in Table 5.5 & 5.6

Table 5.5: The 10-fold classification accuracy and input layer sizes when performing feature selection from -200 Hz to 200 Hz and -100 Hz to 100 Hz

	-200 Hz/200 Hz		-100 Hz/100 Hz	
Encoder thresholds	**Input neurons**	**Accuracy**	**Input neurons**	**Accuracy**
4	3200	53.38 ± 3.57 %	1600	48.72 ± 2.80 %
6	4800	60.97 ± 2.51 %	2400	57.00 ± 4.92 %
8	6400	61.19 ± 5.13 %	3200	58.94 ± 4.71 %
10	8000	62.69 ± 4.02 %	4000	60.38 ± 2.57 %
12	9600	59.31 ± 6.13 %	4800	58.88 ± 3.48 %

Table 5.6: The 10-fold classification accuracy and input layer sizes when performing feature selection from -50 Hz to 50 Hz and -25 Hz to 25 Hz

	-50 Hz/50 Hz		-25 Hz/25 Hz	
Encoder thresholds	**Input neurons**	**Accuracy**	**Input neurons**	**Accuracy**
4	800	40.75 ± 4.09 %	400	39.09 ± 4.43 %
6	1200	50.22 ± 3.71 %	600	45.44 ± 3.49 %
8	1600	51.13 ± 3.64 %	800	46.50 ± 3.29 %
10	2000	51.56 ± 5.29 %	1000	45.16 ± 4.93 %
12	2400	53.75 ± 3.88 %	1200	49.63 ± 4.27 %

As expected the feature selection scenarios where no significant drop in performance can be observed is are between -200 Hz and 200 Hz and between -100 Hz and 100 Hz. Surprisingly, when features are selected between -200 Hz and 200 Hz, the overall performance seems to slightly increase with respect to using all the features. For the other scenarios dropping significant feature directly translates to a deterioration of the classification accuracy. Depending on which aspect of the accuracy-hardware resources trade off is given more weight, determines the "optimal" configuration. If the highest possible accuracy is desired, 10 thresholds with feature selected between -200 Hz and 200 Hz, seems to be the best . If hardware resources need to be limited, 6 thresholds with features selected between -100 Hz and 100 Hz will be a good fit. As this configuration is still within reasonable bounds of the highest possible accuracy, while reducing the amount of input neurons significant. However, optimal is a subjective term as it depends on the target hardware platform and the constraint imposed by the engineer.

Let's further investigate the impact of the feature reduction by looking at the confusion matrices of two configurations, shown in Tables 5.7 and 5.8. There doesn't seem to be huge difference between the two, but there are some shifts in the amount of misclassification. However, overall quite some samples seem to elicit no response. Possible explanations could be that the training hasn't completely converged yet and more epochs are required or Tempotron can't find a combination of weights to map these samples to the correct prediction. To see if more epochs are required, we repeat the experiment.

Table 5.7: The confusion matrix of 6 thresholds with all features of fold 4. Here the X column denotes a no response

		Predicted class				
		Wave	Pinch	Swipe	Click	X
True class	Wave	29	0	6	1	11
	Pinch	0	22	2	1	22
	Swipe	1	2	32	2	9
	Click	0	1	1	30	14

Table 5.8: The confusion matrix of 6 thresholds with features between -100Hz and 100Hz of fold 4. Here the X column denotes a no response

		Predicted class				
		Wave	Pinch	Swipe	Click	X
True class	Wave	30	3	4	2	8
	Pinch	1	25	6	1	14
	Swipe	4	1	34	1	6
	Click	0	6	3	23	15

The results of the additional experiments are presented in Table 5.9. It seems that running the training longer doesn't show any improvement. Meaning it seems that the results we have found now is the maximal achievable performance of the classifier.

Table 5.9: The experiment results for fold 4 running for longer epochs, with features selected between -100 Hz and 100 Hz and using 6 thresholds

Epochs	**Accuracy**
150	55.67 ± 5.97 %
200	55.69 ± 5.15 %

5.5 Self Organizing Map & SNN classifier

In the previous section we saw the performance limits of the SNN classifier. In the experiments with the SOM-LSTM architecture we saw an increase in accuracy when reducing the data dimensionality with a SOM. Wondering if this effect would also be observed in the SNN classifier, we trained the SNN classifier on the output spikes coming from a trained SOM mapping. The parameters shown in Table 5.10 are assumed for the networks.

Table 5.10: Parameters for the SOM-SNN architecture. The synaptic weights are initialized following a uniform distribution between -1 fC and 1 fC, here the uniform distribution is denoted as U

Parameter	Value
Tempotron layer size	4 neurons
Initial synaptic weights	U(-1e-15, 1e15) C
Epochs Tempotron	100
Learning rate Tempotron	5e-17
Epochs SOM	50
Learning rate SOM	0.1

Figure 5.8, shows the 10-fold good unique classification accuracy of the Tempotron classifier for varying SOM grid sizes. A similar trend we've seen for the SOM-LSTM architecture, increasing grid size comes with an increase in classification accuracy. For even larger grid sizes this gain seems to diminish and stabilize. Interestingly, the standard deviation seems to be lower for all grid sizes w.r.t SOM-LSTM. No other SOM grid sizes were explored as memory requirements were too large for the computing platform. It's expected that the performance will increase more, but eventually will converge.

The confusion matrix in Table 5.11 shows the classification results for fold 4 with a 18x18 SOM grid. It seems that performs reasonable on the classes, except for the *Pinch* class which seems to have quite some no responses. Compared to the classification results of the Tempotron classifier (Table 5.7), the amount of correct classifications has remained stable. While the SOM-SNN architecture shows more miss classifications and less non-responses, particularly for the Wave and Click class.

Table 5.11: The confusion matrix of a 18x18 SOM grid for fold 4. Here the X column denotes a no response

		Predicted class				
		Wave	Pinch	Swipe	Click	X
True class	Wave	30	3	8	4	2
	Pinch	1	19	2	4	21
	Swipe	2	4	34	1	5
	Click	1	6	3	30	6

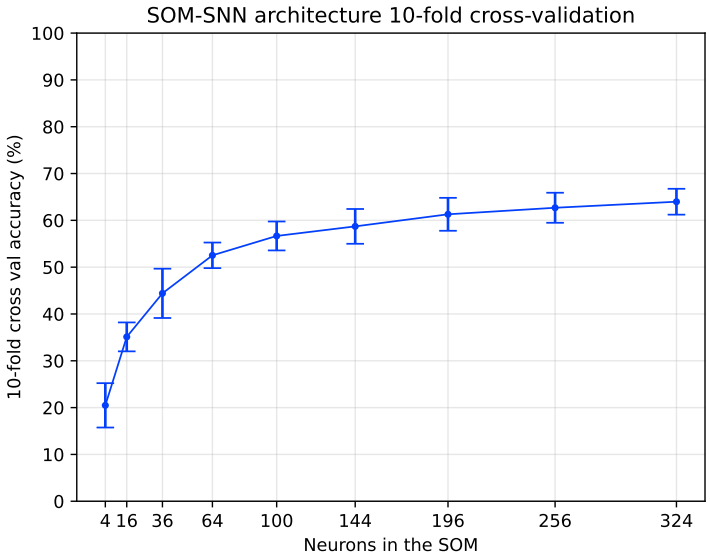


Figure 5.8: SNN 10-fold stratified cross validation classification accuracy for varying SOM grid sizes.For an numerical overview of the results see Table A.3

Compared to the accuracy achieved by the Tempotron classifier, the SOM-SNN architecture seems to achieve the same level of performance at $\backsim$ 64 %. This is somewhat unexpected as the SOM-LSTM improved on the LSTM classifier with 10 %. This raises the question whether this is the highest possible accuracy of the Tempotron classifier on DopNet. Another possibility could be that the current learned SOM mapping doesn't sufficiently separate the samples in the lower dimensional space. If that's the case, further study in the quality of the mapping is required.

It's worth to note that the same level of accuracy has been achieved at a lower spiking neuron cost w.r.t the Tempotron classifier. However, since this is an ANN-SNN hybrid system it could come at the cost of a higher overall power consumption.

We experimented with feature selection on the Tempotron classifier, to combat the increasing amount of neurons in the input layer. This isn't necessary an issue for the SOM-SNN as the input neurons are now determined by the size of the SOM grid. However,it would be interesting to see the effect of feature selection on the SOM's ability to separate observations in the output space. The same feature selection scenarios are assumed, see Figure 5.7. It's likely that a similar trend can be observed to the feature selection experiments in the Tempotron classifier. The 10 fold cross validation accuracy for varying SOM sizes and the feature selection scenarios is shown in Figure 5.9.

As expected the scenarios where no important features were omitted exhibit the same level of accuracy w.r.t using all features. This confirms the same trend seen earlier in the Tempotron classifier.

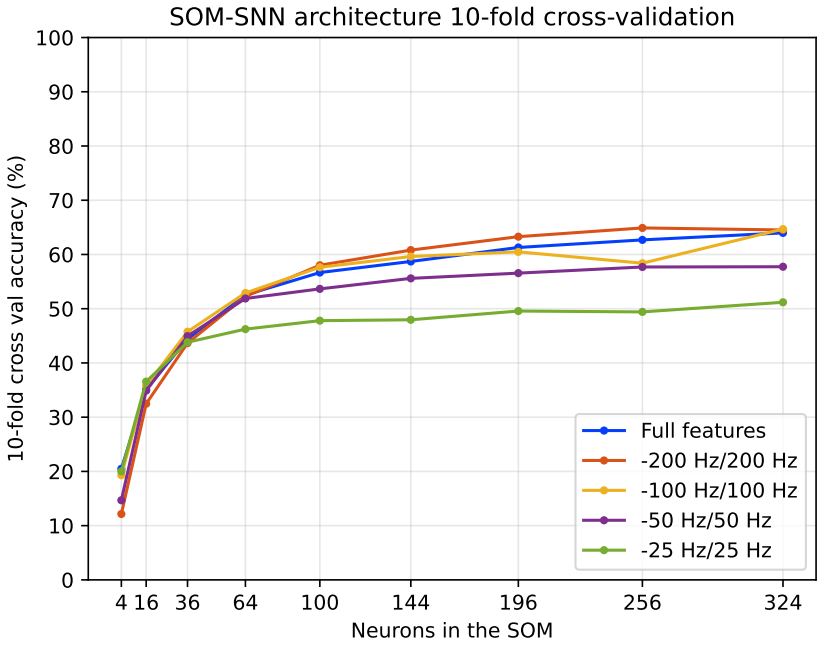


Figure 5.9: SNN feature selected 10-fold stratified cross validation classification accuracy for varying SOM grid sizes. The error bars are omitted for visualization purposes. For an numerical overview of the results see Tables A.3 to A.7

5.6 Spiking Self Organizing Map & SNN classifier

The introduction of the spiking Self Organizing Map (SSOM) opens up a large design space which needs to be explored. By using the Spike Timing Dependent Plasticity (STDP) learning method an additional four parameters are added. A grid search approach will be employed to try to find a point in our design space that can perform the classification we desire. To limit the search space, we will assume the configuration of the encoder to be as presented in Table 5.12 and will not be part of the design exploration. Also, no cross validation will be performed in the experiments presented here, as it was deemed too time consuming to do it effectively. Therefore training and testing is done with fold 4 of our cross validation split, see Section 5.2

Table 5.12: Assumed parameters of the input encoder

Parameter	Value
Encoder thresholds	6
Features selected	-100 Hz/100 Hz
Input layer size	2400 neurons

In all previous experiments the classification accuracy was the metric dictating the performance of our model. Since the SSOM transforms the input data to a lower dimensionality space, the quality of this transformation needs to be assessed. Otherwise, low quality mappings can create misleading accuracy metrics at the SNN classifier due to overfitting.

Metric multidimensional scaling [51, 52] is a technique that maps points from a high dimensional space to a lower dimensional space, such that the distance between each pair of points in the space matches as close as possible given by a set of dissimilarities. A mapping is found that minimizes the error:

$$E_{MDS} = \sum_{i=1} \sum_{j<i} (F(i,j) - G(M(i), M(j)))^2 \tag{5.1}$$

where F and G are the dissimilarity matrix in the higher dimensional space and the dissimilarity matrix after mapping M respectively. For a perfect mapping this error should be zero, as (dis)similar samples in the original space should have the same (dis)similarity in the mapped space.

Common distance metrics used for spike trains are the Victor-Purpura [53] and the van Rossum [54] distance. However, it's been debated by [55], if these metrics are able to asses timing information for higher spike rates. The authors advises against the use of the previously mentioned metrics and instead proposes their own metrics. One of them is the SPIKE-distance, it has the advantages of being parameter free, time scale adaptive and a Python implementation already exists [56]. These advantages have lead to the choice of using this metric in the calculation of E_{MDS}.

We'll be sweeping across values for the initial weights, the STDP parameters and the neighbourhood function as we expect those parameters will have the biggest influence on how the SSOM responses to its input spikes. Sweeping across any parameter is a time intensive task. A single simulation run of the complete SSOM-SNN architecture can take up to an hour to finish. Making it impossible to sweep over all possible values. Therefore, exploratory simulations were performed in order to find the acceptable bound of the parameters. Within those bounds our parameter sweep will take place, the results of which will be discussed here.

We assume the baseline configuration for the SSOM shown in Table 5.14 and the parameters for the SNN classifier as presented in Table 5.13. Recall from Section 4.3 that the all-to-all connections in the SSOM grid are scaled using the Mexican hat function, which has the function call; mexicanhat(α,β, r). With the inhibitory magnitude (α), the inhibitory spread (β), radius of the neighbourhood (r).

This baseline configuration is based on a series of assumptions and exploration experiments. A 20x20 grid is assumed, mainly due to observation from the experiments with the SOM, that larger grids tend to result in higher classification accuracy, likely due to a better spatial separability in the grid. The initial weights and maximum allowed weight size were found by testing various combinations of weights in the exploration experiments. This also true for the STDP learning amplitudes. To limit the chance of overfitting the amount of epochs was set at two. Since STDP can update the weight during cycle of incoming spikes, only a small amount repetitions are needed. The STDP learning windows need to be smaller than the rate at which spikes are entering our classifier. Otherwise, multiple presynaptic spikes can be present in the same learning window. Leading to multiple potentiations and depressions during a single training window, causing instability in the learning process. Since spikes are entering each 8 μs , the learning windows were set at 6 μs. The neighbourhood function was found in [57], where the SSOM we use was first introduced. Lastly, T_{ref} was set at 20 μs in order to prevent neurons in the SSOM from being greedy and fire each time input spikes enter at the input.

Table 5.13: Parameters for the SNN classifier trained by Tempotron, identical to the one used in section 5.4

Parameter	Value
Initial synaptic weights	U(-1e-15, 1e15) C
Epochs	100
learning rate	5e-17
τ_m	0.1 ms
τ_s	0.33 μs
T_{ref}	0 s

Table 5.14: The baseline parameters for the SSOM at the start of the grid search.

Parameter	Value
Grid size	400 neurons (20x20)
Initial weights SSOM input layer	U(2e-16, 6e-16) C
Max weight	3 fC
Epochs SSOM	2
STDP A_{2+}	1.5e-17
STDP A_{2-}	3e-17
STDP τ_+	6 μs
STDP τ_-	6 μs
neighbourhood function	mexicanhat(4,3,2.5)
T_{ref}	20 μs

STDP changes the synaptic weights based on the timing between the pre-and post-synaptic spikes. It can only update the weights if a postsynaptic spike has been elicited. Also, due to it's unsupervised nature, STDP will continue to perform weight updates until all epochs have finished. Extra care should be taken when choosing the initial synaptic weights. If they are too low, not a single neuron will spike. The network will forever be silent as it will never receive any weight updates. If they are too high, all neurons will always fire and no sensible mapping is achieved.

First, we sweep over various values for the lower and upper bound of the uniform distribution that set the initial weights. The sweeps values and their corresponding metrics are shown in Figure 5.10. Here the metrics are presented in a coloured matrix, where the colour is linked to the performance metrics. A good model (low E_{MDS} and high accuracy) will be coloured in green, a bad model (high E_{MDS} and low accuracy) will be coloured red. Models that perform in between these two extremes will have shaded red or green colours.

The overall performance of this subspace is not yet on the same level as seen earlier with the SNN classifier in Section 5.4. However, it seems that for higher values of the maximum initial weight the accuracy improves as does E_{MDS}. Also, a smaller interval of the distribution performs better. It seems as though the values in this subspace are too low, since the best metrics are at the boundaries of this space. For now we'll move to explore another parameter, however it might be worthwhile to revisit the initial weight parameters at some point.

We increase the initial weight distribution from U(2e-16, 6e-16) C to U(4e-16, 25e-16) C for the next sweeps.

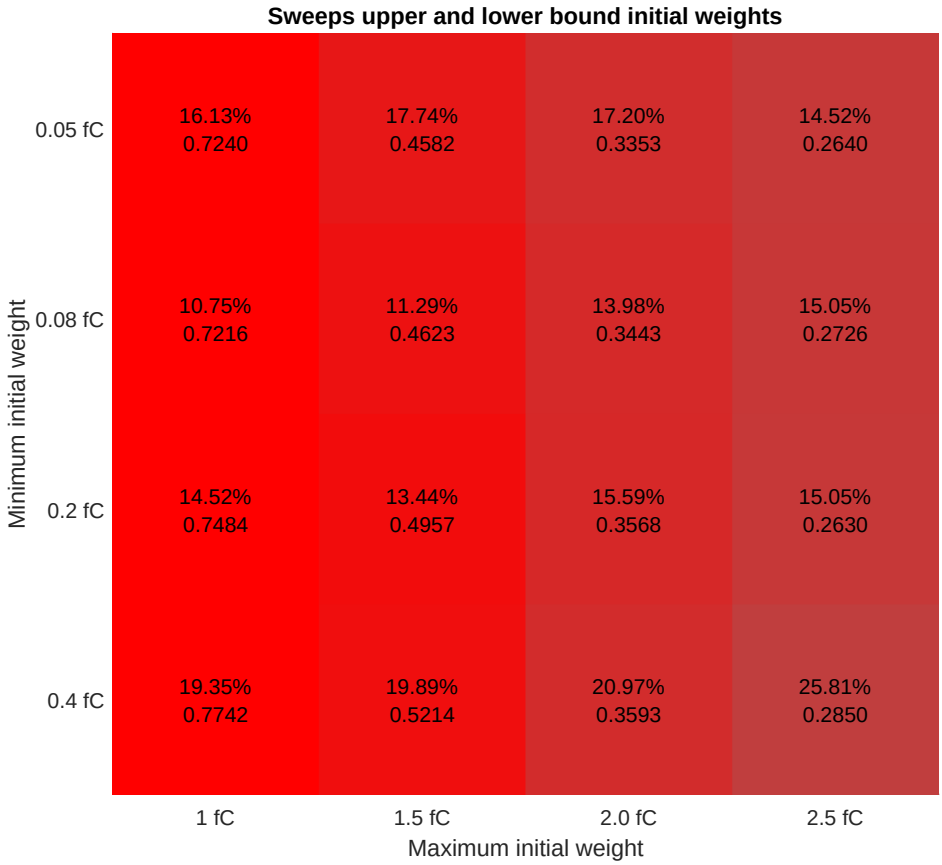


Figure 5.10: The classification accuracy and E_{MDS} for sweeps of the lower(U_{min}) and upper(U_{max}) bound of the initial weights

Next, the potentiation (A_{2-}) and depression (A_{2+}) STDP amplitudes are explored. The resulting metrics are shown in Figure 5.11. It shows that A_{2-} needs to be larger than A_{2+}, otherwise the potentiations will overshadow the depression resulting in constant firing of neurons. On the other end, A_{2-} should not be too large, as this would result in too much depression, silencing the network. Thus, the amplitudes should be relative close to each other. Two configuration in this subspace stand out, one achieving 28,49 % , the other 29,03 %. One might argue that the one achieving 28,49 %, performs better as its E_{MDS} decreases more than the loss in accuracy with respect to the other one. However, we choose to change A_{2+} to 0.035fC and A_{2-} to 0.04fC regardless. Because the overall performance is still low, we choose to go for the higher accuracy. Since the difference in E_{MDS} is marginal, this choice seems acceptable.

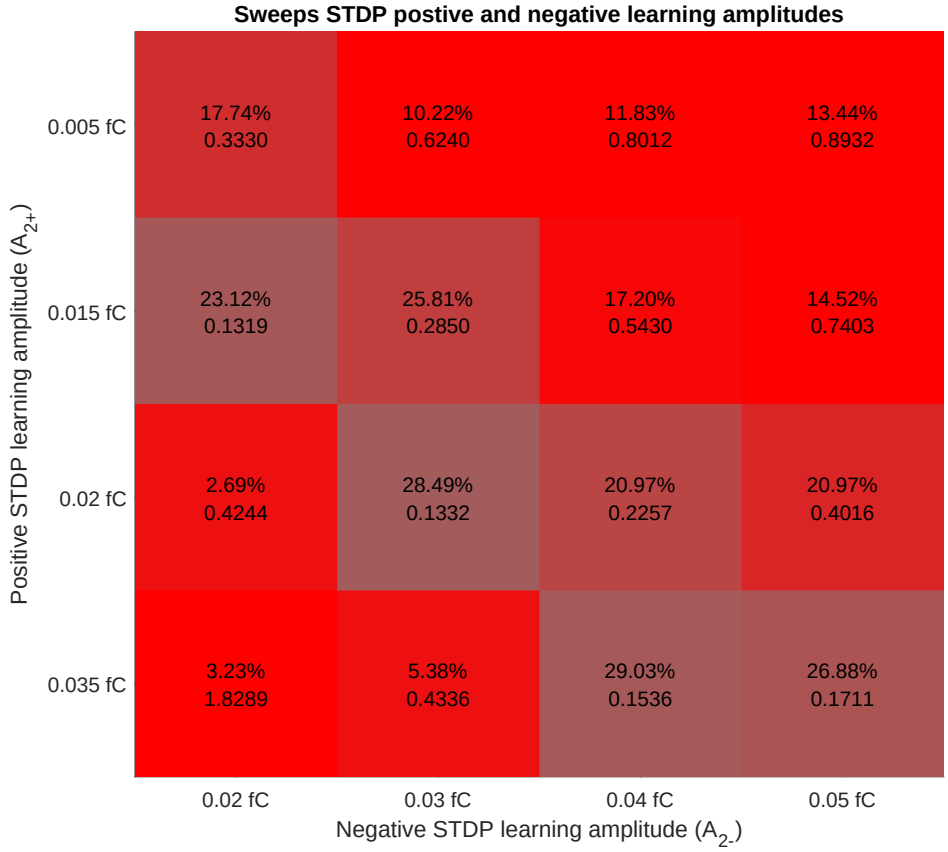


Figure 5.11: The classification accuracy and E_{MDS} for sweeps of A_{2-} and A_{2+}

The sweep results for smaller and larger STDP learning windows are shown in Figure 5.12. A clear relation can been seen between τ_+ and τ_-. Keeping the windows equal seems to make the most difference. Interestingly, a larger τ_+ with respect to τ_- seems to exhibit very low accuracies. We don't update the values for τ_+ and τ_- as no significant improvement on the current values was found.

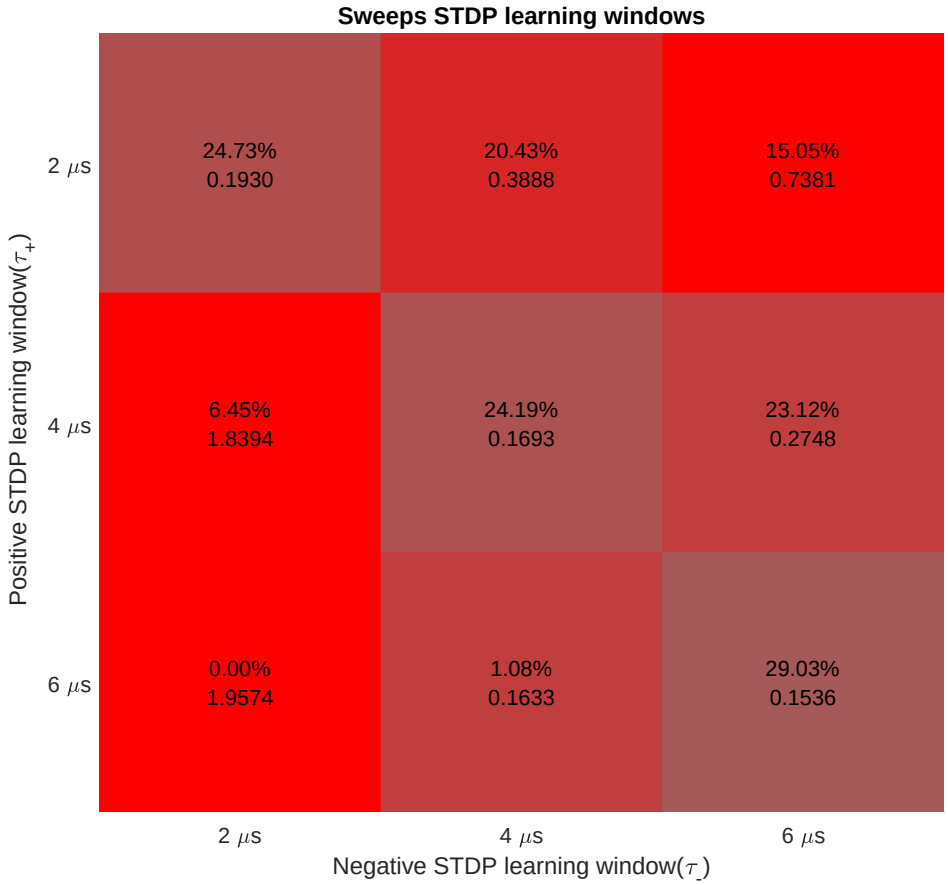


Figure 5.12: The classification accuracy and E_{MDS} for sweeps of τ_+ and τ_-

At the start of the exploration we tested various uniform distributions for the initial weights. Recall that the sweeped values were estimated to be too low. Therefore, another exploration was done for the initial weight distribution with higher weight values. Also, we increased the maximum weight limit from 3 fC to 4 fC. The results from this sweep is illustrated in Figure 5.13. As expected, the weights we explored earlier were on the low side as the metrics in this subspace score overall better with respect to Figure 5.10. The best model in this space achieves an accuracy of 31.72 % and a E_{MDS} of 0.129. We update the initial weight distribution from the earlier found U(4e-16, 25e-16) C to U(8e-16, 15e-16).

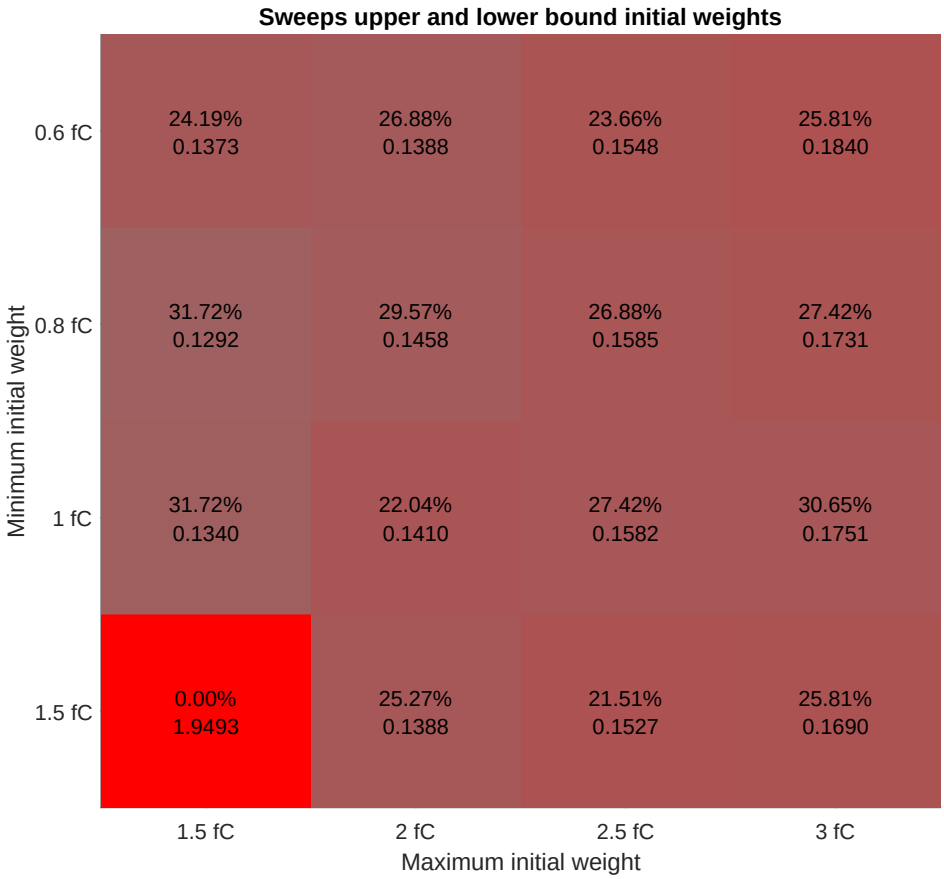


Figure 5.13: The classification accuracy and E_{MDS} for sweeps of the lower(U_{min}) and upper(U_{max}) bound of the initial weights, now with higher weight values

As the neighbourhood function has a major role in determining the mapping, it will be the final parameter we'll explore. We'll gradually increase each parameter of the mexican hat function to explore the behaviour of the neighbourhood function. The sweep results are presented in Figure 5.14. Using a larger radius (r) slightly decreases the classification accuracy but doesn't seem to affect E_{MDS}. Increasing the inhibitory spread (β), negatively affects both the accuracy and E_{MDS}. Interestingly, at $\beta = 5.0$ the accuracy drops with respect to the baseline value. But for larger spreads the accuracy start increasing, while E_{MDS} starts deteriorating. This could be a sign of overfitting taking place. Similar behaviour can be observed for the inhibitory magnitude (α).

Within this subspace no improvement with respect to the baseline values for the neighbourhood function were found. It appears that increasing β and α results in overfitting. This is somewhat expected as neighbouring neuron are inhibited more allowing for the winner neuron to spike more often and become more greedy.

It's important to note that we only explored taking higher values for β, α and r. Therefore, it's quite likely that a better fit for the neighbourhood function could exist, as there are a lot of different untested neighbourhood shapes.

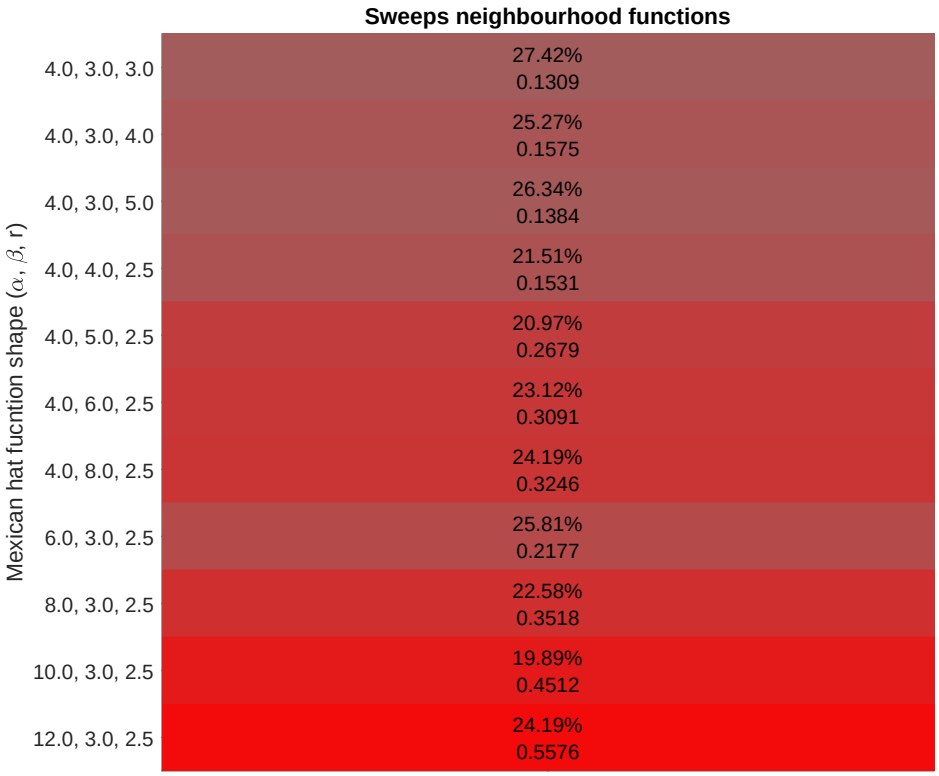


Figure 5.14: The classification accuracy and E_{MDS} for sweeps of neighbourhood function shapes

After the grid search, the SSOM configuration presented in Table 5.15 was found. The configuration we found still lacks in terms of classification accuracy with respect to the SNN classifier. Currently th SSOM makes the input data less spatial separable, effectively making it harder for the classifier to classify the samples. This is the complete opposite what we aimed to achieve by extending the SNN classifier with the SSOM. We could have explored more in the design space to possibly find a better fit for the SSOM, but it wasn't possible due to time constraints on the project.

By inspecting the confusion matrix of the final configuration we can try to explain the lack in performance, it's presented in Table 5.16. Clearly the architecture overfits on the *Click* class. To some extent it's able to distinguish the *Wave* and *Swipe* class. However the *Pinch* class is the hardest to classify, often being misclassified or no responses at all. The amount of no responses is somewhat high but it's at similar level to the no responses shown in Table 5.7. This effectively tells us the mapping in the SSOM does not separate the spike trains in the lower dimensional space enough. Sparking a lot of misclassification and no responses in the SNN classifier. This is can have a variety of causes, most likely due to combination of the initial weights distribution and the neighbourhood function we've chosen. As they have a significant influence on how the mapping is structured in the SSOM. More exploration is required to determine if a better configuration exists in the design space or that the SSOM just isn't able to make create a lower dimensional space where spike trains are spatial separable.

Table 5.15: The final configuration for the SSOM found after the grid search

Parameter	Value
Grid size	400 neurons (20x20)
Initial weights SSOM input layer	U(8e-16, 15e-16) C
Max weight	4 fC
Epochs SSOM	2
STDP A_{2+}	3.5e-17
STDP A_{2-}	4e-17
STDP τ_+	6 μs
STDP τ_-	6 μs
neighbourhood function	mexicanhat(4,3,2.5)
T_{ref}	20 μs

Table 5.16: The confusion matrix of the SSOM-SNN architecture after the grid search and using fold 4 as the test set. Here the X column denotes a no response

		Predicted class				
		Wave	Pinch	Swipe	Click	X
True class	Wave	17	1	8	10	12
	Pinch	4	9	2	11	22
	Swipe	7	4	14	8	13
	Click	4	1	3	22	18

6 Conclusion

This work presents a SNN classifier for hand gesture recognition on Micro-Doppler signatures from the DopNet dataset. To train the synaptic weights of the SNN we employed the Tempotron learning method. This supervised learning method limits the network to a singular layer. Since single layer networks tend to perform worse in terms of classification accuracy, as they aren't able to make complex approximations of the desired mapping function. In order to improve the accuracy, an extension to the classifier was proposed. A spiking version of the Self Organizing Map, capable of transform spike trains from high dimensional space to a lower dimensional space.

Because there was no literature on SNN based solutions for hand gesture recognition, it was unclear if the proposed networks would result in state of the art accuracies. Therefore, two baseline architectures were implemented to estimate the behaviour and limitations of the proposed architectures. First, a LSTM classifier was used during the initial exploration of the DopNet dataset. Testing for varying sizes of the LSTM layer, the maximal achievable accuracy was $\sim$ 74 %. Second, the LSTM classifier was extended with a non spiking SOM to test if dimensionality reduction would increase the performance of the LSTM classifier. An increase in classification accuracy of $\sim$ 10 % could be observed for sufficiently large SOM grids. This confirmed that dimensionality reduction for the LSTM network has a positive effect on the networks ability to generalize on the data.

The SNN classifier was tested for a varying amount of encoder thresholds. We found that the 10 thresholds achieved the highest accuracy at 58.81 %. However, the input encoder had a big impact on the amount of input neurons. Leaving us to explore four feature selection scenarios in order ot limit the input layer size. As expected two of the scenarios achieved similar levels of maximal accuracy at 62.69 % and 60.38 %. The other two showed lower levels of accuracy, because important spatial features were discarded. This showed the trade off between accuracy and network size.

The SNN classifier was also extended with a non spiking SOM to test the dimension reduction hypothesis on the SNN classifier before implementing a spiking SOM. The SOM-SNN architecture showed a marginal improvement on the SNN classifier. A somewhat surprising find, as this not in line with the improvement seen in the SOM-LSTM architecture. Either 64 % is the highest achievable accuracy for the SNN classifier or the SOM mapping's quality isn't sufficient, which would require further study.

With the introduction of the spiking SOM (SSOM) opened up a large design space. Due to the unsupervised nature of the STDP learning methods, searching in this space can be tricky. Therefore, a grid search approach was used to find a functional operating point in the design space. However, during our search it was not possible to solely use classification accuracy as our performance metric. Because this would mask configurations that exhibit overfitting. By adding an additional metric, which assessed the quality of the SSOM mapping we gained additional insights into the behaviour of the

SSOM. Which helped understand how it contributed to the performance of the whole architecture.

Parameters sweeps were performed on the initial weight distribution, STDP learning windows, STDP potentiation and depression amplitudes and different shapes for the neighbourhood function. Each exploration of the subspaces showed us relations between pairs of parameters and their effect on the performance of the SSOM-SNN architecture. With each sweep a marginal improvement on the performance metrics was achieved. Resulting in a configuration with a classification accuracy of 31.72 % and a E_{MDS} of 0.1292. The found model performed worse than the SNN classifier and shows signs of overfitting on the *Click* class. Because we were unable to explore more of the design space it's unclear if there exists another point in the design space that would outperform the current found model. Or that the SSOM implementation used in this work is unable to effectively transform the DopNet data to a lower dimensional space while keeping the data spatially separate. Further study involving the SSOM in this whole pipeline is required.

7 Future Work

The fact that there was no published literature on hand gesture recognition using Spiking Neural Networks at the start of the project shows the novelty of the problem. Assumption were made, based on works that solve similar problems but gave no guarantee it would produce the desired result. For example the choice for the encoding algorithm was based on the assumption audio spectrogram data could be interchangeable with radar spectrogram data. This assumption was correct, but at the cost of more hardware resources compared to the audio application in the literature. This work should therefore be seen as a stepping stone for future work in hand gesture recognition using Spiking Neural Networks, as there is still a lot to gain in terms of classification accuracy and knowledge how these systems operate. Due the time constraint imposed by project not all ideas could be explored, but can be worth pursuing in future works.

The ANN networks main purpose was to provide a first insight in the capabilities of DopNet. For both the LSTM and the SOM training is performed for 50 epochs without any early stop criterion. This could lead to overfitting if training is done for longer then it's actually needed. Even though the results achieved by these networks don't show constant overfitting, implementing an early stopping criterion based on a validation set could be beneficial in lowering the chance of overfitting and achieving higher accuracy. This also holds for the training with Tempotron, as currently the SNN simulator does not support this functionality.

When using a clustering or mapping method, a validation of the clustering or mapping is required. Otherwise, how do you know the effectiveness of the mapping? Normally SOMs are validated via a visual representation, due to added time dimension in our data this becomes more difficult. A numerical validation would be a better solution similar to the approach used in the SSOM design space exploration with the multidimensional scaling error. Currently no such validation is done for the SOM. The performance of the SOM-LSTM and a visually inspection of SOM's output was used as validation, which is not a robust validation method. Implementing the multidimensional scaling error for the SOM would be a meaningful improvement.

Due to size DopNet, going through each individual sample to verify it's quality was deemed too time consuming. Any noisy sample or mislabelled might be part of the training set, affecting the systems performance. Doing any quality test on DopNet either visually or metric based would be useful as it will be quite likely it will improve the classification accuracy.

According to [58], Tempotron does not guarantee to converge to a solution for multi-class problems, due to the discontinuous nature of the cost function. This work assumes this solution can be found. In the same literature a solution is also proposed to solve for this issue, it would be worthwhile to compared the proposed solution with the original Tempotron on a multi-class problems. Also recently, [59] proposed a hardware compatible multi-layer SNN trained with an extended version of Tempotron called;*DeepTempo*.

As this method might solve the single layer limitation imposed by Tempotron while remaining hardware friendly, might be worthwhile to investigate.

The motivation to use the spiking SOM implementation by [24], was mainly due to it's availability in the SNN simulator to avoid intensive implementation and testing process. As discussed early in Section 4.3, the synaptic weights of the synapses connecting the grid neurons are scaled using a neighbourhood function referred to as the "Mexican Hat" function. This scaling is done during initialization and does not change over time. For a normal SOM the neighbourhood function shrinks over time, to ensure a converges of the clusters. Since the neighbourhood function has a central role in the self organization process, looking into the behaviour more different functions might help in understanding the organization behaviour of the spiking SOM. Also, due to the nature of the grid search approach and the long simulation times limited us to explore only a certain part of the whole design space. Applying a different hyperparameter optimization method might be useful in further optimizing the SSOM. Preferably an automated process, perhaps a gradient-based method could help in exploring the design space faster.

Furthermore, as neuromorphic hardware platforms mature, the hardware parameters assumed in this work might no longer hold. Understanding how the proposed system acts with different hardware parameters is essential if it's to be used in real world application. This is not explored in depth as we only assume specific hardware parameters.

The synapse model in the SNN simulator assumes spikes are instantaneous events. Therefore, it was required to set the time constant $\tau_s \approx 0s$ for the Tempotron learning method. This could have had an impact on the final performance of the SNN classifier. Implementing a different synapse model that can produce more complex postsynaptic potentials might be worth pursuing as it might have a positive effect on the performance of the Tempotron trained classifier.

The simulations done in this work don't take into account any noise generated by the radar sensor or the hardware. In a real world application, the model needs to operated when exposed to certain levels of noisy conditions. Therefore a more in depth examination of the models robustness to noise seems necessary when deploying the model in a real world environment.

Bibliography

[1] P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, D. C. May, and G. Brain, "TensorFlow : A system for large-scale machine learning,"

[2] A. Paszke, S. Gross, J. Bradbury, Z. Lin, Z. Devito, F. Massa, B. Steiner, T. Killeen, and E. Yang, "PyTorch : An Imperative Style , High-Performance Deep Learning Library," no. NeurIPS, 2019.

[3] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, and H. Raja, "Soli : Ubiquitous Gesture Sensing with Millimeter Wave Radar," vol. 142, no. July, 2016.

[4] M. Ritchie, R. Capraru, and F. Fioranelli, "DopNet: A micro-Doppler radar data challenge," *Electronics Letters*, vol. 56, no. 11, pp. 568–570, 2020.

[5] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neuroscience*, vol. 9, no. 3, pp. 420–428, 2006.

[6] I. J. Tsang, F. Corradi, M. Sifalakis, W. Van Leekwijck, and S. Latré, "Radar-based hand gesture recognition using spiking neural networks," *Electronics (Switzerland)*, vol. 10, no. 12, pp. 1–20, 2021.

[7] B. Yin, F. Corradi, and S. M. Bohte, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," pp. 1–15, 2021.

[8] D. Banerjee, S. Rani, A. M. George, A. Chowdhury, S. Dey, A. Mukherjee, T. Chakravarty, and A. Pal, "Application of Spiking Neural Networks for Action Recognition from Radar Data," *Proceedings of the International Joint Conference on Neural Networks*, 2020s.

[9] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," *UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 851–860, 2016.

[10] E. Hayashi, J. Lien, N. Gillian, L. Giusti, D. Weber, J. Yamanaka, L. Bedal, and I. Poupyrev, "Radarnet: Efficient gesture recognition technique utilizing a miniature radar sensor," *Conference on Human Factors in Computing Systems - Proceedings*, 2021.

[11] S. Skaria, A. Al-Hourani, M. Lech, and R. J. Evans, "Hand-Gesture Recognition Using Two-Antenna Doppler Radar with Deep Convolutional Neural Networks," *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3041–3048, 2019.

[12] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10336–10346, 2021.

[13] S. Hazra and A. Santra, "Short-Range Radar-Based Gesture Recognition System Using 3D CNN with Triplet Loss," *IEEE Access*, vol. 7, pp. 125623–125633, 2019.

[14] C. Liu, Y. Li, D. Ao, and H. Tian, "Spectrum-Based Hand Gesture Recognition Using Millimeter-Wave Radar Parameter Measurements," *IEEE Access*, vol. 7, pp. 1–13, 2019.

[15] K. Academy, "Multi polar neuron." https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/the-synapse, 2021. Accessed: 12.08.2021.

[16] M. authors, "Schematic of an action potential." https://commons.wikimedia.org/wiki/File:Action_potential.svg, 2020. Accessed: 20.07.2021.

[17] F. Jülic and SeitenPlan, "Wiry synapse." https://effzett.fz-juelich.de/en/2-19/wiry-synapse/?print=1, 2019. Accessed: 20.07.2021.

[18] Frank Rosenblatt, "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

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

[21] A. F. H. A. L. Hodgkin, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[22] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *J. Physiol. Pathol. Gen*, vol. 9, pp. 620 – 635, 1907.

[23] N. Brunel and M. C. Van Rossum, "Quantitative investigations of electrical nerve excitation treated as polarization," *Biological Cybernetics*, vol. 97, no. 5-6, pp. 341–349, 2007.

[24] J. Mes, E. Stienstra, X. You, S. S. Kumar, A. Zjajo, C. Galuzzi, and R. Van Leuken, "Neuromorphic self-organizing map design for classification of bioelectric-timescale signals," *Proceedings - 2017 17th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2017*, vol. 2018-Janua, pp. 113–120, 2018.

[25] R. Brette, "Philosophy of the spike: Rate-based vs. Spike-based theories of the brain," *Frontiers in Systems Neuroscience*, vol. 9, no. November, pp. 1–14, 2015.

[26] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, no. DEC, pp. 1–12, 2017.

[27] Z. Liang, D. Schwartz, G. Ditzler, and O. O. Koyluoglu, "The impact of encoding–decoding schemes and weight normalization in spiking neural networks," *Neural Networks*, vol. 108, pp. 365–378, 2018.

[28] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.

[29] J. P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *Journal of Neuroscience*, vol. 26, no. 38, pp. 9673–9682, 2006.

[30] T. Kohonen and S. Member, "The Self-Organizing Map," vol. 78, no. 9, pp. 1464–1480, 1990.

[31] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[32] I. Valles-Perez, "Self-organizing maps, the kohonen's algorithm explained." https://ivape3.blogs.uv.es/2015/03/15/self-organizing-maps-the-kohonens-algorithm-explained/, 2015. Accessed: 20.07.2021.

[33] H. Hazan, D. Saunders, D. T. Sanghavi, H. Siegelmann, and R. Kozma, "Unsupervised learning with self-organizing spiking neural networks," *arXiv*, 2018.

[34] B. Ruf and M. Schmitt, "Unsupervised learning in networks of spiking neurons using temporal coding," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1327, no. September, pp. 362–366, 1997.

[35] B. Ruf and M. Schmitt, "Self-organization of spiking neurons using action potential timing," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 575–578, 1998.

[36] M. Richards, J. A. Scheer, and W. Holm, *Principles of Modern Radar: Volume I - Basic Principles*. 2010.

[37] V. C. Chen, F. Li, S. S. Ho, and H. Wechsler, "Analysis of micro-Doppler signatures," *IEE Proceedings: Radar, Sonar and Navigation*, vol. 150, no. 4, pp. 271–276, 2003.

[38] V. C. Chen, F. Li, S. S. Ho, and H. Wechsler, "Micro-doppler effect in radar: Phenomenon, model, and simulation study," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 2–21, 2006.

[39] J. H. H. Mark S. Zediker, Robert R. Rice, "United States Patent (19) 11 Patent Number : BATTERY-49," *United States Patent*, no. 5,847,817, 1998.

[40] T. M. Inc., "Introduction to micro-doppler effects." https://www.mathworks.com/help/radar/ug/introduction-to-micro-doppler-effects.html;jsessionid=336adae416495c210494f5a8461e, 2019. Accessed: 26.07.2021.

[41] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand gestures recognition using radar sensors for human-computer-interaction: A review," *Remote Sensing*, vol. 13, no. 3, pp. 1–24, 2021.

[42] M. Ritchie and A. M. Jones, "Micro-doppler gesture recognition using doppler, time and range based features," *2019 IEEE Radar Conference, RadarConf 2019*, 2019.

[43] Z. Pan, J. Wu, M. Zhang, H. Li, and Y. Chua, "Neural population coding for effective temporal classification," *arXiv*, 2019.

[44] S. Hochreiter and J. Schmidhuber

[45] G. Chevalier, "LARNN: Linear Attention Recurrent Neural Network," 2018.

[46] C. Pehle and J. E. Pedersen, "Norse - A deep learning library for spiking neural networks," Jan. 2021. Documentation: https://norse.ai/docs/.

[47] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[48] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith, "Nengo: A Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. JAN, pp. 1–13, 2014.

[49] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, pp. 1–41, 2019.

[50] D. Spessot, "Design space exploration of a spiking lsm classifier for radar applications," Master's thesis, Delft University of Technology, Dec. 2019.

[51] W. S. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[52] G. J. Goodhill and T. J. Sejnowski, "A Unifying Objective Function for Topographic Mappings," *Neural Computation*, vol. 9, no. 6, pp. 1291–1303, 1997.

[53] J. D. Victor and K. P. Purpura, "Nature and precision of temporal coding in visual cortex: A metric- space analysis," *Journal of Neurophysiology*, vol. 76, no. 2, pp. 1310–1326, 1996.

[54] M. C. W. V. Rossum, "Communicated by Jonathan Victor," *Neural Computation*, vol. 763, no. June, pp. 751–763, 2001.

[55] T. Kreuz, D. Chicharro, M. Greschner, and R. G. Andrzejak, "Time-resolved and time-scale adaptive measures of spike train synchrony," *Journal of Neuroscience Methods*, vol. 195, no. 1, pp. 92–106, 2011.

[56] M. Mulansky and T. Kreuz, "PySpike—A Python library for analyzing spike train synchrony," *SoftwareX*, vol. 5, pp. 183–189, 2016.

[57] J. Mes, "Design space exploration of a neuromorphic ecg classi

cation system using a spiking self-organizing map," Master's thesis, Delft University of Technology, Sept. 2018.

[58] R. Urbanczik and W. Senn, "A gradient learning rule for the tempotron," *Neural Computation*, vol. 21, no. 2, pp. 340–352, 2009.

[59] "DeepTempo: A Hardware-Friendly Direct Feedback Alignment Multi-Layer Tempotron Learning Rule for Deep Spiking Neural Networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1581–1585, 2021.

Appendix

A

A.1 Experiment results LSTM classifier

Table A.1: 10-fold cross validation classification accuracy of the LSTM classifier for varying LSTM sizes

LSTM cells	Accuracy
1	45.07 ± 7.26 %
2	62.07 ± 9.28 %
4	66.25 ± 5.42 %
8	69.52 ± 3.42 %
12	71.83 ± 4.55 %
16	71.19 ± 4.66 %
20	72.74 ± 4.68 %
24	73.39 ± 4.37 %
28	72.81 ± 3.38 %
32	72.25 ± 2.34 %
36	73.82 ± 3.33 %
40	74.14 ± 4.26 %
44	74.52 ± 3.30 %
48	75.32 ± 4.39 %
52	76.28 ± 3.53 %
56	73.12 ± 3.38 %
60	73.40 ± 4.28 %
64	73.11 ± 2.67 %

A.2 Experiment results SOM-LSTM architecture

Table A.2: 10-fold cross validation classification accuracy of the SOM-LSTM architecture for varying SOM grid sizes

SOM dimensions	Grid neurons	Accuracy
2x2	4	53.29 ± 14.72 %
4x4	16	61.00 ± 8.64 %
6x6	36	64.26 ± 7.15 %
8x8	64	67.48 ± 14.05 %
10x10	100	70.24 ± 10.2 %
12x12	144	65.46 ± 10.86 %
14x14	196	82.14 ± 6.41 %
16x16	256	83.21 ± 2.84 %
18x18	324	80.86 ± 5.33 %

A.3 Experiments results SOM-SNN architecture

Table A.3: 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using all features

SOM dimensions	SNN input layer size	Accuracy
2x2	4	20.49 ± 4.74 %
4x4	16	35.11 ± 3.09 %
6x6	36	44.41 ± 5.26 %
8x8	64	52.53 ± 2.73 %
10x10	100	56.67 ± 3.09 %
12x12	144	58.71 ± 3.72 %
14x14	196	61.29 ± 3.52 %
16x16	256	62.88 ± 3.21 %
18x18	324	63.98 ± 2.76 %

Table A.4: 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -200 Hz and 200 Hz

SOM dimensions	SNN input layer size	Accuracy
2x2	4	12.15 ± 4.32%
4x4	16	32.47 ± 5.70 %
6x6	36	43.65 ± 4.96 %
8x8	64	52.26 ± 4.22 %
10x10	100	58.01 ± 4.38 %
12x12	144	60.81 ± 3.87 %
14x14	196	63.28 ± 3.42 %
16x16	256	64.89 ± 2.99 %
18x18	324	64.52 ± 4.49 %

Table A.5: 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -100 Hz and 100 Hz

SOM dimensions	SNN input layer size	Accuracy
2x2	4	19.30 ± 5.89 %
4x4	16	35.81 ± 4.57 %
6x6	36	45.75 ± 6.00 %
8x8	64	52.90 ± 4.81 %
10x10	100	57.64 ± 4.86 %
12x12	144	59.62 ± 3.19 %
14x14	196	60.48 ± 3.80 %
16x16	256	58.39 ± 3.33 %
18x18	324	64.68 ± 2.34 %

Table A.6: 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -50 Hz and 50 Hz

SOM dimensions	SNN input layer size	Accuracy
2x2	4	14.68 ± 5.94 %
4x4	16	34.89 ± 3.73 %
6x6	36	44.95 ± 4.41 %
8x8	64	51.88 ± 4.76 %
10x10	100	53.66 ± 4.27 %
12x12	144	55.59 ± 3.01 %
14x14	196	56.56 ± 2.66 %
16x16	256	57.56 ± 2.85 %
18x18	324	57.74 ± 2.43 %

Table A.7: 10-fold cross validation classification accuracy of the SOM-SNN architecture for varying SOM grid sizes and using features between -25 Hz and 25 Hz

SOM dimensions	SNN input layer size	Accuracy
2x2	4	20.00 ± 5.36 %
4x4	16	36.56 ± 5.43 %
6x6	36	43.82 ± 4.25 %
8x8	64	46.24 ± 2.33 %
10x10	100	47.80 ± 4.07 %
12x12	144	47.96 ± 2.98 %
14x14	196	49.57 ± 1.89 %
16x16	256	49.41 ± 4.26 %
18x18	324	51.18 ± 2.80 %