BACHELOR

Spiking Neural Networks for gesture recognition in WiFi CSI sensing a feasibility study

Luteijn, Sophie C.A.

Award date:
2023

Awarding institution:
Tilburg University

Department of Mathematics and Computer Science
Interconnected Resource-Aware Intelligent Systems

Spiking Neural Networks for gesture recognition in WiFi CSI sensing: a feasibility study

Bachelor Thesis

S.C.A. Luteijn

Supervisors:
Prof.Dr.Ir. N. Meratnia
MSc B.R.D. van Berlo

Eindhoven, January 2023

Abstract

WiFi CSI sensing has yielded promising results in activity recognition. Different machine learning models have been studied in the context of human gesture recognition, but they have demonstrated to be domain sensitive. Spiking neural networks could be capable of addressing the domain shift problem, but first the in-domain performance must be evaluated. In this thesis spiking neural networks with different spike encodings and LIF neuron models are evaluated for the SignFi and MNIST datasets. We also compare the performance of spiking neural networks to the performance of convolutional neural networks. The conclusion is that, under certain conditions, it is feasible to use spiking neural networks for gesture recognition in WiFi CSI sensing, but that more research is needed to understand the impact of the data and the different network components on the training process and the in-domain performance.

Keywords*: WiFi sensing, channel state information, spiking neural network, convolutional spiking neural network, activity recognition, gesture recognition.*

Contents

Chapter 1

Introduction

As human computer interaction becomes more prevalent in daily life, there is an increase in the use of human activity recognition in numerous fields such as smart homes, security surveillance, healthcare and entertainment [54]. Different sensing technologies are used to monitor activities and machine learning algorithms are used to detect, recognize and estimate expressions of motion, including human non-verbal communications [34, 38].

A distinction can be made between device-based sensing techniques and device-free sensing techniques. Device-based sensing techniques require the user to carry a device to collect data. Carrying a device might work for some applications, but can also be very impractical. Device-free sensing techniques are used to monitor the environment. Commonly known device-free sensing techniques include visual and audio data. While such methods make data collection easier, they also have several drawbacks. For instance, the use of techniques based on video or audio is rather intrusive as subjects can be identified from the data. In addition, for visual sensing, a subject needs to be located in line-of-sight without occluding obstacles in the view and there are requirements regarding the environment, such as the lighting conditions [68].

Ma, Zhou & Wang state that Wireless Fidelity (WiFi) sensing with Channel State Information (CSI) data is not intrusive and easy to implement because WiFi networks are already widely deployed. Channel State Information is a three-dimensional matrix that stores information on the time-varying amplitude attenuation and phase shift of radio signals that are sent between transmitter and receiver [10, 33, 61]. The receiver does not only receive signals from the line of sight (LoS) path but also signals from transmission, reflection, refraction, diffraction, adsorption and scattering [64]. Activities that are carried out between or close to the transmitters and receivers cause signal disruptions [41]. These disruptions generate amplitude attenuation and phase shift. CSI can be used to calculate the size, speed, and location of the disruption [60].

One of the main assumptions is that there exists a fixed mapping between human activity and CSI signal features [65]. Each signal variation pattern is associated with a particular activity. Machine learning can be used to recognize human gestures based on the extracted features [10, 64]. However, there is also a problem that we are still facing. Changes in the domain have an impact on the performance of activity recognition systems. This issue is known as the domain shift problem [39]. In this case the domain can be defined as the set of all domain factors. These domain factors include but are not

limited to, the spatial environment of the experiment, the location and orientation of the person performing the activity, from now on referred to as the performer, the physical properties of the performer, the velocity with which a gesture is performed and even the time of day. Changes in any of the domain factors can impact the signals and degrade the performance of recognition models [54, 66, 68].

There are two main strategies academics have used to address the domain shift problem. The first method is to create more generalized models [5, 39]. The other more popular method is to use domain adaptation [10, 13, 59].

1.1 Problem Statement

Various machine learning and deep learning frameworks have already demonstrated that they can provide some answer to the domain shift problem. However, there are several strategies that have not been fully evaluated in the context of gesture identification using WiFi CSI data. The use of Spiking Neural Networks is one machine learning model that should be investigated further. Before any conclusions can be drawn about the domain shift resilience of SNNs, we first must evaluate its in-domain performance in the context of WiFi CSI.

1.2 Research Questions

The purpose of this thesis is to evaluate the effect on the in-domain performance of different spiking neural network components for gesture recognition on the SignFi dataset and the MNIST dataset. The main research question is as follows: *How well does a spiking neural network perform in in-domain WiFi CSI gesture recognition with different component settings?* To answer this question, three subquestions are formulated:

- How do different rate and latency spike input encodings influence the performance of a spiking neural network in gesture recognition?
- Which variation of a leaky integrate and fire (LIF) neuron performs best for gesture detection on WiFi CSI?
- How does a spiking convolutional neural network perform compared to a convolutional neural network?

1.3 Project Structure

Beginning with the introduction in Chapter 1, the motivation, problem and questions are discussed. Then in Chapter 2 background information is provided to explain the concept of WiFi CSI and the different methods that are used in Chapters 3, 4, 5 and 6. In Chapter 3, a brief literature review is given on the use of the techniques described in Chapter 2, from which the research questions in Chapter 1 originate. In the methodology in Chapter 4 an explanation is given of the pipeline and how the research questions are answered. In

Chapter 5 an explanation is given of the experiments conducted according to 4 and results are analyzed and discussed. Chapter 6 summarizes the results from Chapters 4 and 5, how these answer the research questions presented in Chapter 1 and how this relates to the information in Chapter 3.

Chapter 2

Background Information

This chapter provides background information on WiFi CSI, DFS, convolutional neural networks and spiking neural networks, and some components of spiking neural networks.

2.1 WiFi CSI

In WiFi CSI sensing, a transmitter sends a radio signal using an antenna to a receiver. These radio signals are electro-magnetic waves [26]. WiFi CSI utilizes the Multiple-Input Multiple-Output technique (MIMO). This means that multiple antennas are used at the transmitter and receiver. While combining information from different antennas reduces errors, it also enables the data to travel over multiple paths at the same time [18, 33]. The radio waves carry information on a carrier frequency signal along these paths. At the same time orthogonal frequency-division multiplexing (OFDM) is used to split the carrier frequencies into different subcarrier frequencies. This way multiple data streams can be transmitted in parallel [32, 68]. Different paths have different effects on the attenuation, which is the reduction of the amplitude, and the phase shift of the subcarrier frequency [62]. The data from the different subcarriers, or channels, can be combined. The resulting signal, which is a superimposition of the different channels over the different paths, can be measured at the receiver in voltage. Converting the voltage measurement to decibel results in the received signal strength indicator (RSSI) [56].

Where RSSI only gives an indication of the cumulative signal strength, the CSI also contains information on the different channels [5]. The CSI data at a single moment is represented in a matrix $H^{N\times M\times K}$ with N receiving antennas and M transmitting antennas [33, 62, 68]. The number of subcarriers is represented by K. Including the fourth dimension, the time domain denoted as T, results in the matrix $H^{N\times M\times K\times T}$ as shown in figure 2.1. The CSI for each moment in time is calculated as follows:

$$H(f,t) = \sum_{n}^{N} a_n(t) \exp^{-j2\pi f \tau(t)}$$

N represents the total number of paths, $a_n(t)$ the complex attenuation factor, $t_n(t)$ the propagation delay and f the frequency. In an ideal scenario, the received signal would be the result of multiplying the transmitted signal with the CSI matrix. However, random

phase errors are caused by packet boundary detection, sampling frequency offset, carrier frequency offset and an unknown initial phase [29, 62]. This yields in a received signal for each subcarrier that can be modeled by $y = Hx + n$. Where y is the received signal, H the CSI, x the transmitted signal and n the noise.

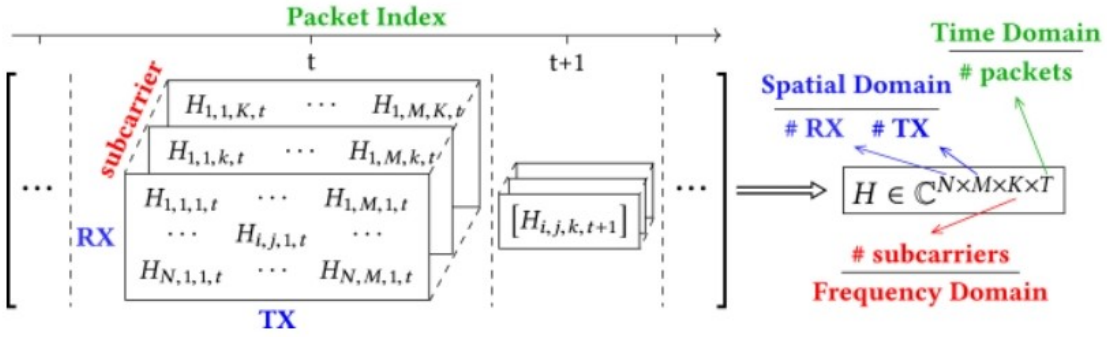

Figure 2.1: Four dimensional CSI tensor containing the multipath channel variations, image taken from [33].

2.1.1 DFS

The CSI data described above can be preprocessed further. The frequency domain consists of multiple subcarriers K. As we are interested in the differences in the environment, we want to look at the subcarrier which provides the most information. A principal component analysis (PCA) is used to find the subcarrier with the most significant changes over time. The resulting data is in the form $H^{N \times M \times 1 \times T}$.

A short-time Fourier transform (STFT) is then performed on the time domain to generate the DFS spectrum. Next, the principal value amplitude is obtained [34, 62, 66]. Lastly, the information on the spatial domain $N \times M$ is stacked into one dimension. The resulting DFS data is in the form $DFS \in R^{N \times B \times T}$ with N representing the spatial domain, B the frequency bins and T the time.

2.2 Deep Learning Algorithms

Artificial neural networks (ANN) are computational models that are inspired by the powerful and complex structure of the human brain. In the human brain, neurons are cells that pass electrical impulses and chemical signals to other neurons [28, 44]. In deep learning, artificial neurons, or nodes, are used to mimic the electrical activity of the brain. The nodes are connected to other nodes by synapses. These synapses represent small gaps. The node has an associated weight and a threshold function is used to determine if the node in the next layer is activated [2, 9]. The information is given to the neural network via the input layer. The input layer is followed by one or more hidden layers and an output layer. Artificial neural networks can be trained for better performance.

2.2.1 Convolutional Neural Network

Convolutional neural networks are artificial neural networks primarily used for pattern recognition in images and computer vision [2, 40]. Convolutional neural networks reuse a set of weights, also known as the filter or kernel, on the output values in a layer [50]. The kernel shifts over the image performing element-wise multiplication with the part of the image over which the kernel is hovering [1, 40]. This results in reduction in the needed preprocessing compared to other artificial neural networks. A convolutional neural network is able to capture spatial and temporal dependencies and consists of different layers. These layers are convolutional layers, pooling layers and fully connected layers [27, 40]. The convolutional layer multiplies the input image with the kernel resulting in output neurons. After this a non-linear transformation is applied. The pooling layer is used to downsample the spatial domain and filter out the important information. This is especially needed once the input size of images increases to limit the exponential expansion of the network [40, 50]. The fully connected layers work like the layers in a standard artificial neural network and are used for the classification tasks. Compared to feed forward artificial neural networks, convolutional neural networks are more suitable for recognition tasks using image data.

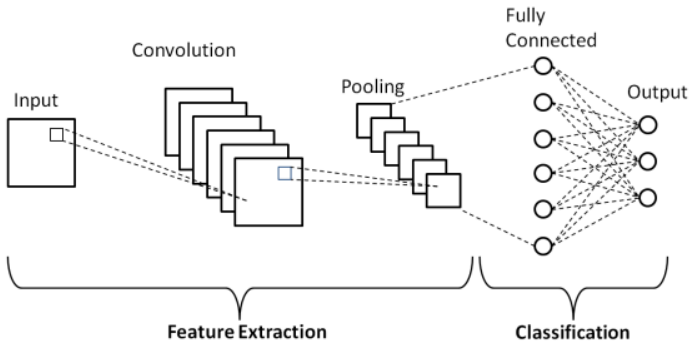

Figure 2.2: Convolutional neural network architecture, image taken from [8].

2.2.2 Spiking Neural Network

The neurons in an artificial neural network are connected by chemical and electrical synapses. The action potentials are the elementary unit of signal transmission. The goal of spiking neural networks is to mimic the electrical synapses in a more biologically realistic way [19, 31]. Natural neural networks are approached by including the synaptic status of neurons and the time. The membrane potential of a neuron increases when a spike is received and then decreases over time. If the membrane potential reaches a certain threshold an impulse is sent from the presynaptic neuron to the postsynaptic neuron and the membrane potential drops [17, 19, 52]. In addition, the human visual system does not process all information at once. Instead, small parts of information are considered at the time. The different inputs received over time lead to the activation of neurons [6]. The most realistic approach requires neuromorphic hardware. Nonetheless, it is possible to convert the data to spike trains and feed this into a neural network [17]. Figure 2.3

depicts a spiking neural network architecture. The input data, in the form of images, is converted into spike trains. After that, the spike trains are fed into the spiking neural network. The synapsis in the human brain is represented by the link between two layers. The model records the output spikes and the membrane potential. This information is transformed back to a numerical class. The membrane potential is recorded in the neuron, every time the membrane potential reaches the threshold, a spike is fired.

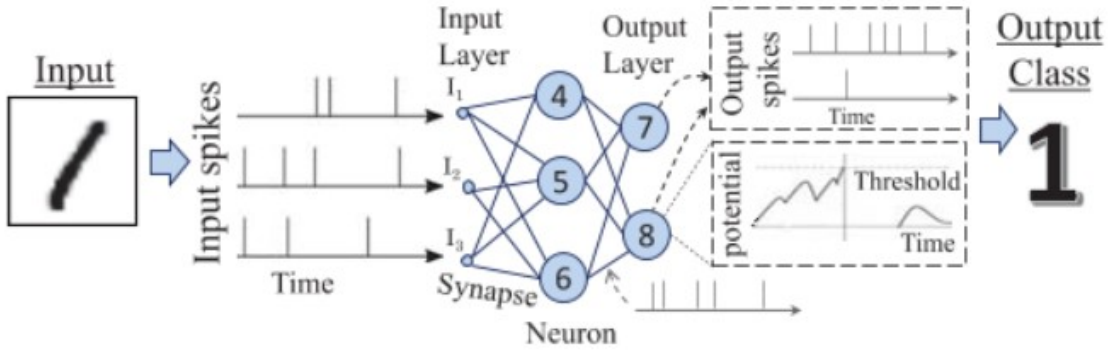

Figure 2.3: Spiking neural network architecture, image taken from [22].

2.2.3 Spike Encodings

This section provides a brief explanation of the encoding techniques that are being evaluated. There are different spike encoding methods all based on the human brain, some more biologically realistic than others and some more complex than others. Two types of spike encoding can be distinguished, rate coding and temporal coding [3]. Rate coding simply measures the number of occuring spikes. Whereas temporal coding can be in the form of time to first spike coding, also known as latency coding, phase coding and burst coding. A representation of the different encoding methods is given in figure 2.4. SnnTorch [17] supports rate coding, latency coding and delta modulation. From these predefined methods, rate coding and latency coding will be evaluated.

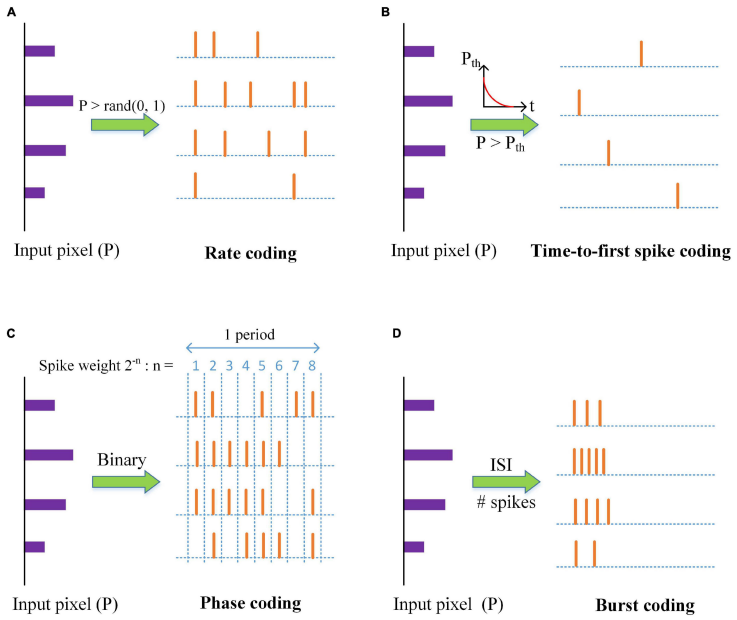

Figure 2.4: An overview of different neural encoding schemes, image taken from [21].

Rate Coding

In rate coding the number of spikes are measured over time. At first the initial input data is repeated along the first dimension [16]. The number of repetitions is defined by the number of time steps, S. Each input feature of the resulting matrix is considered as a probability. The probabilities are used as inputs for a Bernoulli trial where the probability of spike is equal to the input feature. Combining the different trials with different probabilities results in a Poisson spike train.

Latency Coding

In latency coding the time-to-first spike is used. Similar to rate coding, the inputs are repeated for each time step. Except now a LIF neuron is used with a logarithmic decay to determine if there is a spike. For each datapoint in the original input, the time-to-first spike is recorded in dimension S [16].

2.2.4 Neuron Models

Due to the use of electrical potentials, most research on neuron models originates in physics. These models are based on resistor-capacitor (RC) circuits. Based on such RC circuits are the Leaky Integrate-and-Fire (LIF) neuron models [17].

Leaky

The snnTorch package provides the Leaky neuron model. This is a first-order leaky integrate-and-fire model. [17]. The neuron's inner state, also known as the membrane potential, is defined by U. Everytime an input current I is received, the potential U is

increased with I. In addition, the neuron potential decays logarithmically over time. Once the threshold is reached the neuron is either reset to zero or the threshold is subtracted.

Synaptic

The Synaptic model is a second order LIF neuron that also takes into account synaptic conductance. It is similar to the Leaky neuron except that it not only takes into account the membrane potential decay rate beta, but that it also takes the synaptic current, alpha, into account. In the human brain the synaptic decay rate would correspond to synaptic fatique, which causes a neuron to spike less soon after a previous spike [7, 16].

Chapter 3

Related Work

This chapter provides a synopsis of the relevant works on WiFi-CSI gesture recognition that are used as a foundation for this thesis. Next to this, literature is used to explain the choices for the model components of the Spiking Neural Networks used in the experiments.

3.1 Learning Methods

Many researches have been conducted on the use of WiFi CSI data in human activity recognition. Ma Yongsen et al. [33] have provided a comprehensive study of the different methods and applications used in for example human detection, motion detection, spoofing attack detection, activity recognition, motion recognition and gesture recognition.

In WiFall [23] Han et al. have used k-nearest neighbors (kNN) to classify the human movements walk, sit, stand up and fall. In 2014 their framework, designed to help living elders, already achieved 87% precision. Shobehy et al. [48] have utilized kNN classification for indoor localization. A comparison is also made between kNN classification and CNN classification for the respective setup. In this research kNN outperforms the use of CNN in the localization task. The researchers in Arshad et al. present Wi-Chase [3], a Fine kNN classifier that reaches an average accuracy of 94.2% on human activity recognition for 3 gestures. Li et al. [51] mentions how WiFinger, another kNN classifier is able to obtain up to 90.4% average accuracy on finger-grained gesture recognition for 9 gestures. Another tradional machine learning classification technique that has been thoroughly epxlored is the use of support vector machines (SVM). Both kNN algorithms and SVM algortihms are often combined with principal component analysis (PCA) and dynamic time warping (DTW). An example of the use of support vector machines in gesture recognition is WiSign [47]. WiSign employs two receivers. Each receiver predicts the gesture without information from the other receiver. In the end, the predictions are aggregated, and the final prediction is determined by a majority vote. In total 10 different classifiers are trained and majority vote is used over the different classifiers. This results in 93.8% accuracy in the classification of 5 different American Sign Language gestures.

In [34] the researchers mention the very low recognition accuracy in case of an increasing number of classes in gesture recognition with kNN and SVM. This corresponds with how in WiG, DeNum and WiSign [25, 47, 67] the SVM models have achieved high accuracy, 88% NLoS in WiG, 94% in DeNum and 93.8% in WiSign, but only for a very low number of classes being 4 in WiG, 10 in DeNum and 5 in WiSign. The same goes for the kNN models. In WiAG [58], a 94% accuracy was obtained for 6 gestures. [36] resulted in a 92% accuracy for 25 signs and in WiFinger [51] a 90.4% accuracy was obtained for 9 hand postures. In WiMu [57] the researchers have also shown the decrease in accuracy for different numbers of classes in pattern recognition. Where they achieve a 95% accuracy for 2 classes this decreases to 94.6%, 93.6%, 92.6% and 90.9% for 3, 4, 5 and 6 classes.

Other classification algorithms are needed for better gesture recognition once the number of classes increases. Convolutional Neural Networks have already shown to provide a better solution than traditional machine learning algorithms [34]. In Widar3.0 body velocity profiles are derived for cross-domain activity recognition for 15 gestures. The use of kinetic characteristics that are irrespective of the domain lead to 92.7% accuracy in-domain and 92.4% cross-domain accuracy for different environments and 88.9% cross-domain accuracy for different users [66]. In [63] a 93.2% in-domain accuracy was achieved and an 87.1% cross-domain accuracy. Another initiative that aims to reduce the domain dependency in WiFi CSI gesture recognition is SignFi. SignFi feeds pre-processed CSI measurements to a 9-layer CNN for gesture classification [34]. On average SignFi reaches a 94.81% accuracy for 276 sign gestures. The domains include two different environments and five different users.

3.2 SNN Image Classification

One type of learning method that is less explored is the deep learning technique Spiking Neural Networks. As the difficulty of recognition tasks increases, there is a need for more energy efficient implementations. The human brain is the most energy-efficient neuromorphic system there is, making it an appealing source of inspiration. The brain consumes only around 20W, which is nine orders of magnitude lower than computers performing similar tasks [4, 45]. However, increasing accuracy in recognition tasks requires a substantial increase in the size of the network. In [45] an idea is proposed to delete synapses in order to increase hardware efficiency. Another idea is to use Spike-Timing-Dependent-Plasticity (STDP) for training rather than recurrences [30]. Using an unsupervised SNN, a 95% accuracy was obtained on the MNIST benchmark in [15].

As mentioned in section 2.2.2, the input in a Spiking Neural Network consists of spike trains. Rate coding simply measures the number of occuring spikes. Whereas temporal coding can be in the form of time to first spike coding, phase coding and burst coding. According to [21] time to first spike coding, also known as latency coding [17], achieves the highest computational performance. Phase coding, is the most resilient to input noise while burst coding offers the highest robustness to hardware non-efficiency. Burst coding is based on the idea that neurons send a burst of spikes in opposition to phase coding where the spikes are measured over different periods.

When the human body receives a stimuli, there is some time before the brain starts processing the information. This is the foundation for time-to-first spike coding [17].

Eshraghian et al. also mention the benefits of rate and latency coding. One main advantage of latency coding is that it consumes less power. Where for rate coding there are many spikes, for latency coding only the first spike is recorded. Another advantage is the runtime of spike encoding as less spikes need to be processed. An objection to the use of latency coding versus rate coding would be that rate coding captures much more information. More input spikes implies that there are other spikes that can cause a neuron to fire when another neuron fails to do so. Next to this a larger number of spikes results in more information for the model to learn from. This could result in less epochs needed to train a model. In latency coding, it is possible to compensate for having less information by having more neurons. In the human brain a sudden impulse also causes more other neurons to be activated than a constant pulse [37].

In addition to different input encoding schemes, there are also different ways to model the synaptic neurons. In general, the models used so far are variations of the leaky integrate-and-fire neuron. In such models no attempt is made to describe the shape of the action potential. [19]. The leaky integrate-and-fire neuron model is the most elementary spiking neuron model [11]. It has been used widely as a baseline model. Not taken into account in the Leaky neuron is the postdischarge refractoriness as described by Chacron et al. in [11]. Leaky LIF neuron models only use current and future information and do not remember previous spikings. They operate under the assumption that the threshold for firing is constant. More physiologically plausible would be the inclusion of a dynamic threshold. One way to include the previous information in a Leaky LIF neuron would be to subtract the threshold for every spike and to use a threshold that slowly increases. Another option is to include a second decay rate, representing the decay of the synaptic current, alpha. This is the approach taken in the Synaptic LIF neuron model by [17].

Chapter 4

Methodology

To answer the research questions, experiments with different component settings are performed to determine whether spiking neural networks are viable in WiFi CSI gesture recognition under certain conditions. The overarching hypothesis is that spiking neural networks can be used to classify gestures from WiFi CSI data.

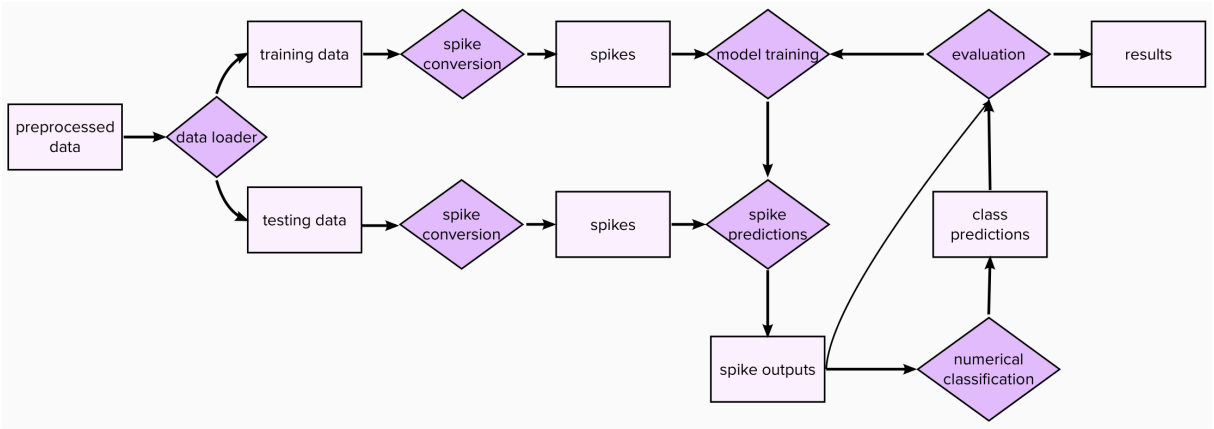

Figure 4.1: A schematic representation of the SNN pipeline.

An overview of the general SNN pipeline is shown in figure 4.1. At first, transformations are applied to the images, such that the images are converted to tensors and normalized. Then datasets are created for the training and testing set and subsets are selected from these datasets. A validation set is used to establish independency of the test set. To iterate over the data, a dataloader is used with batch size 16 and the images are loaded in random order. The data is transformed into spikes for the SNNs. This step is omitted for CNN. The model's training is the following stage. In this stage the average mse loss and accuracy for the training and validation data are returned for each epoch.

For the SNN models a similar structure is used as presented by Eshraghian et al. [16]. The convolutional network architecture used in the MNIST experiments is a 8C5-MP2-24C5-MP2-384FC10 architecture. Where 8C5 is a 5×5 convolutional layer with 8 filters. MP2 is a 2×2 max pooling function. This is followed by a 5×5 convolutional layer with 24 filters and another 2×2 max-pooling function. 384 neurons are then mapped to 10 outputs using a fully connected layer. For the SignFi data a 8C5-MP2-24C5-MP2-

12696FC16 architecture is used. The only distinction between the model architectures is the final fully connected layer. The SNNs contain a leaky neuron layer after each of the two max-pooling layers and after the last linear mapping.

The number of iterations per epoch for the MNIST data is set at 100. The training procedures employ the Adam optimizer, with betas ranging from 0.9 to 0.999 and a constant learning rate at default of 0.001. The best performing model is selected based on the loss per epoch for the training and validation set.

A test set is then used to further evaluate the model. Since the predictions of a spiking neural network are in the form of output spikes, the predictions first need to be converted to numerical predictions as shown in figure 4.2. For each batch the model returns a 3-D matrix $S \times B \times C$ with S time steps, batch size B and C number of classes. For each 2-D matrix $S \times C$ in dimension B, the number of spikes, represented by ones, are counted per class. The class with the most non-zero values is the predicted class. The numerical predictions can be used to construct the confusion matrix and to calculate accuracy, precision, recall and the F1-score.

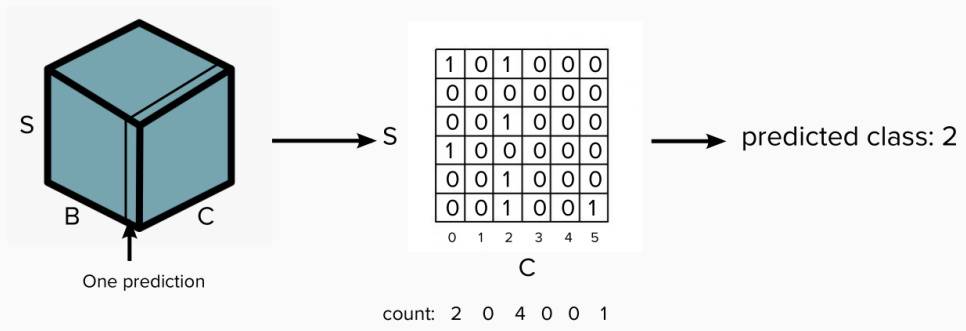

Figure 4.2: Spike classification to numerical classification for 6 classes.

To answer the first subquestion, 12 experiments are conducted to evaluate the impact of rate coding and latency coding with varying time steps and spike threshold settings on the performance of an SNN. Because of the increasing runtime due to increase in the number of steps, the choice is made to evaluate for 30 epochs for MNIST rate coding and for 50 epochs for all other experiments. The different experiment settings are shown in figure 4.3. As the input tensors are not time varying the time varying input boolean variable in rate coding is set to False. This also means that the number of time steps needs to be specified. The spiking neural network is trained for 50, 100 and 250 time steps for both MNIST and SignFi. The input data for a batch before spike conversion is in the form $B \times I$ with B representing the batch size and I the input size. After spike conversion the spike input data is in the form $S \times B \times I$ with S being the specified number of time steps. As increasing the number of time steps increases the spike input size, this also means a massive increase in the runtime. Intuitively, more input spikes are anticipated to result in a better-performing network since there will be more data for the model to train on. Based on human neural network structure, the expectation would be that biologically more realistic temporal latency spike encoding, results in better performance. For latency coding the threshold needs to be optimized in such a way that there is enough information for the model to recognize the gesture but that there is no abundance in spikes resulting

in all images looking similar. The spiking neural network is trained for thresholds 0.01, 0.005 and 0.001. The expectation is that for latency coding, the default setting of 0.01 is also the setting that achieve the best performance.

Experiment	SignFi	MNIST	Conversion	Time steps	Threshold
1	X		rate	50	
2	X		rate	100	
3	X		rate	250	
4	X		latency		0.01
5	X		latency		0.005
6	X		latency		0.001
7		X	rate	50	
8		X	rate	100	
9		X	rate	250	
10		X	latency		0.01
11		X	latency		0.005
12		X	latency		0.001

Figure 4.3: Experiment set for spike input evaluation.

To address the second subquestion, the SNN's Leaky LIF layers will be replaced with Synaptic LIF layers for one fixed spike encoding and only one combination of settings. The alpha is set to 0.85. Both networks are trained for 50 epochs and for both LIF layers the surrogate gradient of the Heaviside step function is used, $spikegrad.surrogate.fast-sigmoid$ [49, 55]. Since an SNN with the Leaky layers has already been trained on the MNIST and SignFi data, just two further tests with the Synaptic layers are required. Multiple metrics will be used to analyze the impact of the various LIF layers, and the training procedure will also be assessed. The experiments written in bold in figure 4.4 are the extra experiments that need to be conducted to answer this subquestion. It is expected that the SNN performs better on the training set with the Synaptic layers than with the Leaky layers because of the inclusion of the biologically realistic synaptic fatigue. However, the Synaptic neuron model might also prevent neurons from firing enough spikes.

Experiment	SignFi	MNIST	Leaky	Synaptic
1	X		X	
2		X	X	
3	**X**			**X**
4		**X**		**X**

Figure 4.4: Experiment set for neuron model evaluation.

To answer the third subquestion, a CNN is constructed using a similar model structure as the SNN without the LIF layers and the data is fed into the CNN without spike conversion. For the MNIST dataset a model is used with a 5×5 convolutional layer with 8 filters followed by a 2×2 max-pooling function after using a ReLU activation function. This step is repeated with a 5×5 convolutional layer with 16 filters. The output is then flattened, after which the outputs are mapped from 256 neurons to 64 neurons to 10

outputs using fully connected layers. For the SignFi dataset the fully connected layers map 8464 neurons to 64 neurons to 16 outputs. The cross-entropy loss function is only applied. The SNN with the highest test set accuracy will be compared to the CNN. The tests needed are presented in figure 4.5. An overview of the CNN pipeline used to answer this subquestion is presented in figure 4.6.

Experiment	SignFi	MNIST	SNN	CNN
1	X		X	
2		X	X	
3	**X**			**X**
4		**X**		**X**

Figure 4.5: Experiment set for comparison between CNN and SNN.

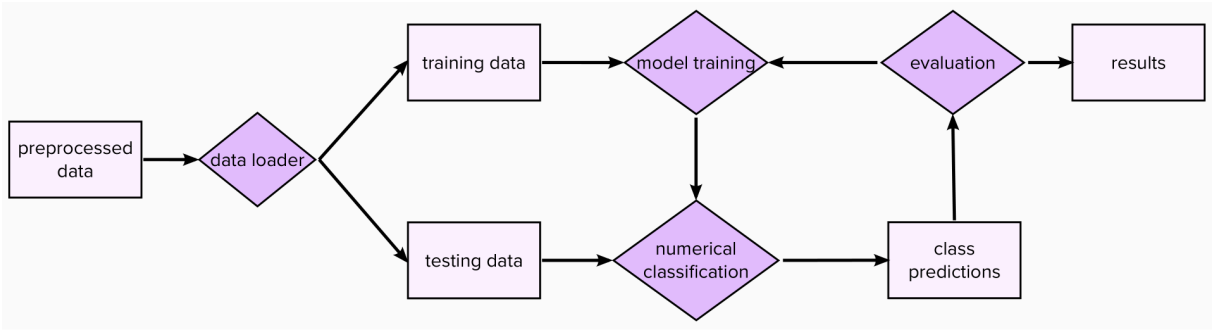

Figure 4.6: Convolutional neural network pipeline.

Chapter 5

Experiments

5.1 Data

There are two open datasets that will be used in this thesis. These are the MNIST dataset [46] and the SignFi dataset [33]. Not all data is utilized directly from the open datasets. This section provides more details on what data is used and how the data is used.

5.1.1 SignFi

The first of the two open datasets used in this thesis is the SignFi dataset collected by researchers at the College of William and Mary. The datasets contain CSI traces and ground truth labels for the different sign gestures [33]. The first dataset contains 8280 entries for 276 sign gestures performed in the lab environment and home environment. Before the classification performance of the models can be evaluated, it first needs to be shown that the data and respective models can be used for classification. In the scope of this thesis, a subset containing the DFS profiles for the first 16 gestures is taken from the lab dataset. A transformation is applied to the data to convert the images to grayscale and to normalize the data. The subset contains 20 images per gesture for each of the 16 gestures. Random stratified sampling is used to create the training, validation and test sets.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Total**
Training	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	160
Validation	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	20
Test	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	20

Figure 5.1: Class representation for SignFi subsets.

5.1.2 MNIST

The second dataset is the MNIST gesture dataset [46]. The original dataset is loaded using torchvision and normalized. The training set contains 60000 images and the test set 10000. The first 2000 instances of the training set are used for training and the next 496 for validation. A subset of 496 instances from the test set is used for testing. As random

sampling is used without stratification, there is a slight class imbalance which should be taken into account in the evaluation.

Class	0	1	2	3	4	5	6	7	8	9	Total
Training	191	220	198	191	214	180	200	224	172	210	2.000
Validation	43	56	48	49	65	46	54	51	50	34	496
Test	42	67	55	45	55	50	43	49	39	51	496

Figure 5.2: Class representation for MNIST subsets.

5.2 Implementation

The PyTorch framework is used to build the experiment models in Python [42]. On a Lenovo Thinkpad P1 Intel Core i7 Gen 8, the CPU is used to run each experiment locally. 15 GB of RAM were available. The snnTorch package is used for the spiking neural networks and the code is based on their available tutorials [16] and the tutorials provided by [20]. Other packages that are used are torch [12], torchvision [35], torchmetrics [14], numpy [24], pandas [53], and Scikit-learn [43]. For optimization, Adam is used with a constant learning rate of 0.001 and the loss is measured using the mean squared error loss and the cross-entropy loss. The runtime of the models varies from less than 5 minutes for the SignFi convolutional neural network to approximately 14 hours for a spiking neural network training loop with 250 time steps rate encoding for MNIST.

5.3 Results

5.3.1 Different Input Encodings

For the SignFi and MNIST datasets, there are 12 distinct tests with three different parameters tested for each input encoding. The training and validation loss are plotted per epoch for each experiment to illustrate how different settings influence the training processes of the SNN models. The logical outcome would have been that the models learn from the training data, at least to some extent. These behaviors are evident in the MNIST plots in figures 5.4 and 5.6. In contrast, for the SignFi dataset, the figures 5.3 and 5.5 show that the models only learn very little and that the model performance does not improve significantly after training.

For SignFi data with rate encoding, the loss does initially drop substantially. After the initial drop the loss stays constant. We do see that for both SignFi and MNIST, the lower the number of steps, the faster the model learns and the better the model performs on the training and validation set. This is surprising because more steps means more information for the model to learn from. For SignFi data with latency encoding, the loss function is similar for each threshold and does not show any improvements. Both the training loss and validation loss show an increase after 11 epochs. This could mean that the learning rate is too large for the model that is used.

Evaluation of the models using the SignFi test set results in 6.25% accuracy for each of the models with rate encoding. To evaluate on the test set, the model at epoch 49 is

chosen. We see the same results for latency encoding. For every separate experiment the model predicts every instance in the test set to be the same gesture, the gesture differs per experiment. For this the model at epoch 27 is chosen.

The models show improvements for both rate conversion and latency conversion for the MNIST dataset. The loss functions show that the model with rate coding learns faster than the model with latency encoding and performs better after 30 epochs. Again, fewer steps result in improved performance in both the training and validation sets. The loss functions for both the training and validation sets show that the model is clearly still improving despite being trained for 50 epochs. For the SNN with MNIST rate encoding, fewer time steps result in smaller loss, as observed in the SignFi data. Another similarity is that the threshold setting in latency encoding has only a minor effect on the loss function. For the MNIST training data, the lower threshold results in slightly lower loss and faster learning. The better performance with a lower threshold is still visible in the validation data, but it is less evident. For 50 steps rate encoding on MNIST, the SNN achieves an accuracy of 96.57% accuracy for 50 epochs on the test set. For 100 steps, the model at epoch 27 achieves an accuracy of 96.37% and for 250 steps the model at epoch 28 achieves an accuracy of 96.98%. For latency coding with threshold 0.01 an accuracy is achieved of 92.14% on the test set by the model at epoch 47. Setting the threshold to 0.005 results in an accuracy of 93.15% for 49 epochs and setting the threshold to 0.001 results leads to an accuracy of 93.35% on the test set after training on 49 epochs.

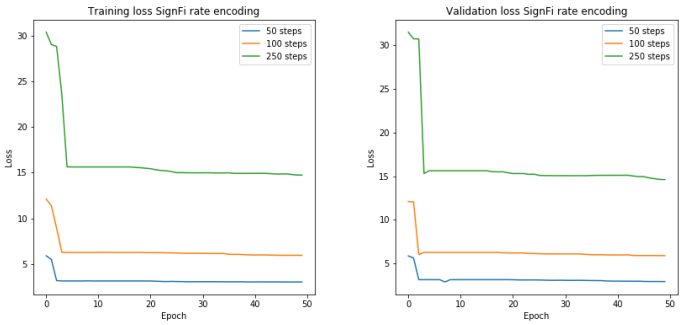

Figure 5.3: Training and validation loss for SignFi rate encodings.

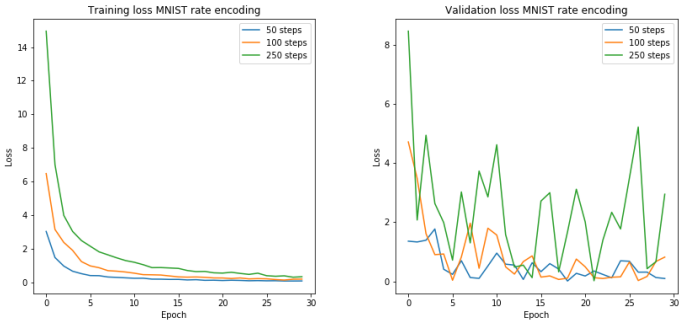

Figure 5.4: Training and validation loss for MNIST rate encodings.

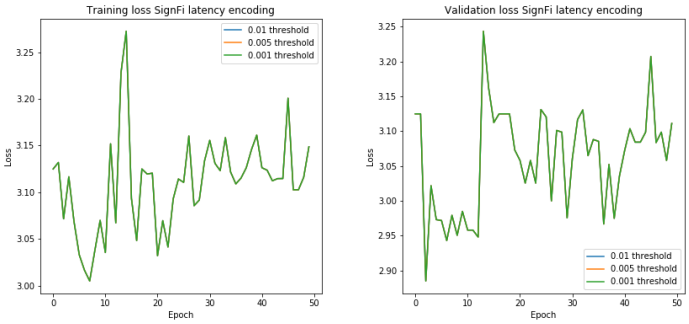

Figure 5.5: Training and validation loss for SignFi latency encodings.

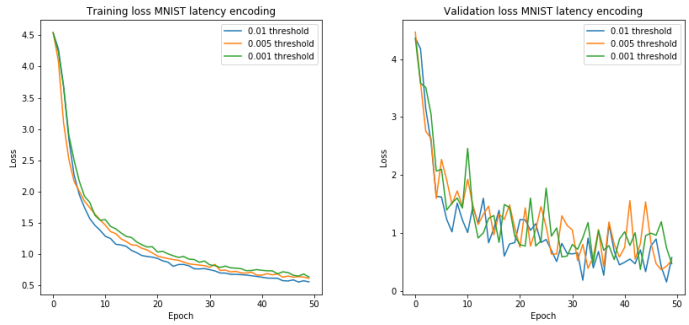

Figure 5.6: Training and validation loss for MNIST latency encodings.

5.3.2 Variations in Neuron Models

For the second subquestion the Leaky LIF neuron in the SNN model is replaced by a Synaptic LIF neuron.

For the SignFi data, the loss drops significantly at first as shown in figure 5.7. After the initial drop the model continues to improve slightly for both the training and validation data but not much. We can see that there is underfit for 50 epochs of training, which could indicate that the model would perform better after longer training. In figure 5.8 the confusion matrix resulting from running the model on the test set is shown. Interesting to see is that the SNN with Synaptic LIF actually yields 28.13% accuracy. Note that most of the false predictions are predicted to be in the first class. In the plot on the left in figure 5.8, the accuracy, precision, recall and F1-score are plotted per class. The boxes in these plots show the second and third quartile. The dots in the plots for precision and F1-score indicate outliers. On the SignFi data, under similar circumstances, the model with the Synaptic LIF performs better than the model with the Leaky LIF neuron model.

For the MNIST dataset the Synaptic neuron SNN is able to achieve a 99.26% accuracy on the training set and a 100% accuracy on the validation set. The loss function shows a very unceasing improvement for the training set, but also shows more variation for the validation set. Evaluating on the test set yields in 96.37% accuracy. In figure 5.9 the evaluation metrics are shown for both the Leaky layers and the Synaptic layers. The results show that the accuracy, precision, recall and harmonic mean are all better for the

SNN with Leaky LIF layers for the experiment conducted. The confusion matrices in figure 5.10 show the actual and predicted classes resulting from running the model on the test set. The plot on the left is the confusion matrix for the SNN with Leaky LIF and the plot on the right is the confusion matri for the SNN with Synaptic LIF. The confusion matrices both show that for the MNIST dataset, the model has learned to an extend where false predictions are spread over all classes.

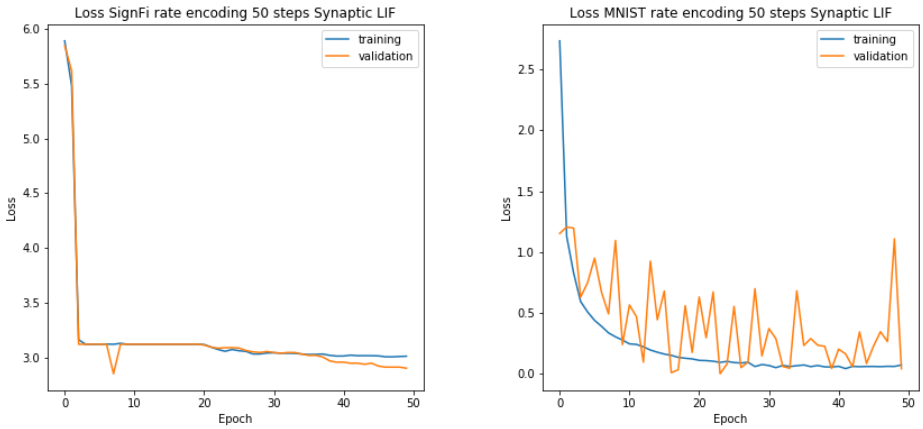

Figure 5.7: SignFi and MNIST loss for Synaptic LIF SNN.

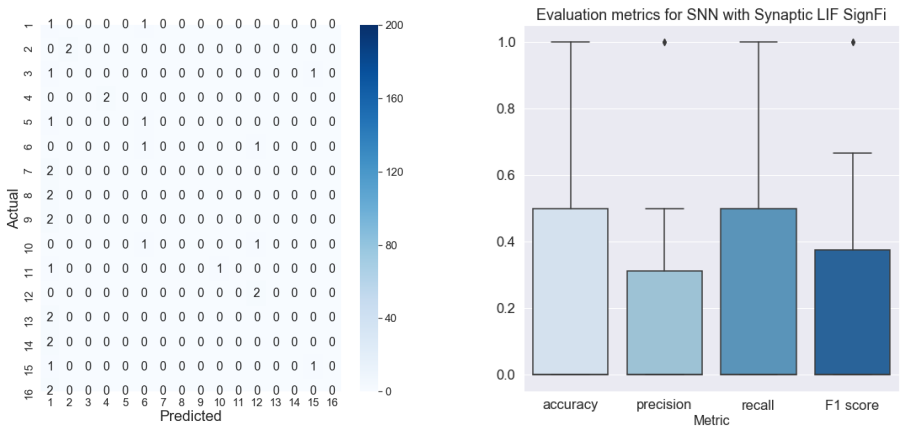

Figure 5.8: Confusion matrix and evaluation metrics for Synaptic LIF neurons in SignFi SNN.

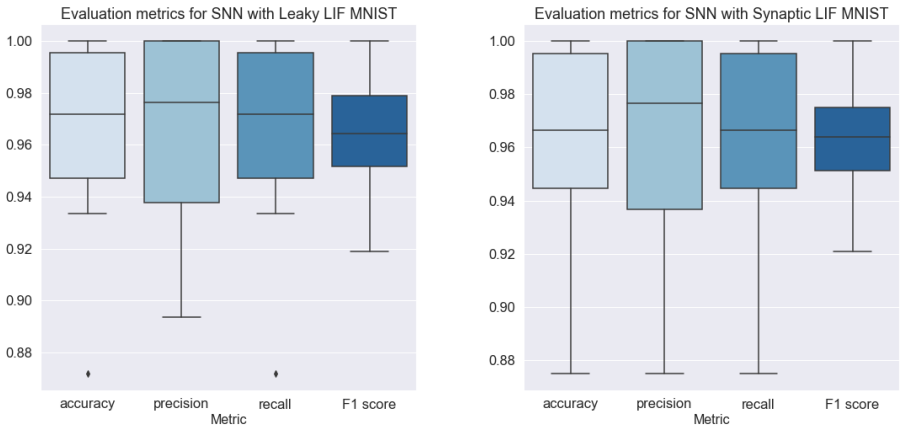

Figure 5.9: Evaluation metrics for Leaky and Synaptic LIF neurons in MNIST SNN.

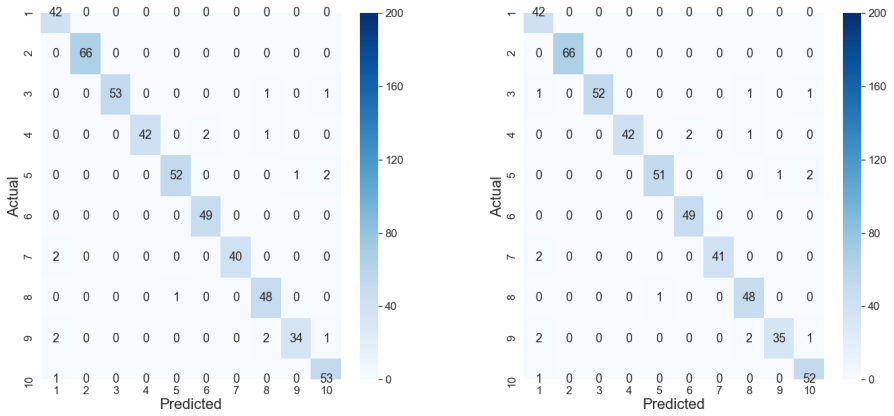

Figure 5.10: Confusion matrices for Leaky and Synaptic LIF neurons in MNIST SNN.

5.3.3 CNN Comparison

The last question to answer is how a spiking neural network compares to a convolutional neural network. For this two CNNs are run for the MNIST and SignFi data.

For the SignFi data, the loss and accuracy functions in figure 5.11 display that the CNN is able to learn from the data but miserably fails to generalize and attain good performance on the training and validation set. Where the model reaches an 97.97% accuracy on the training data. The results for the validation and test set show 0.00% and 6.25% accuracy. For the test set, the CNN predicts everything to be in the same class. This is also what causes the positive outliers for accuracy and recall in the evaluation metrics.

The CNN yields better results for the MNIST data as shown in figure 5.13. The model learns really well on the training data, reaching 100% accuracy. These results are also matched for the validation set. On the test set the model achieves 97.58% accuracy for epoch 48. There is still quite some variation in loss and accuracy for the different

epochs. Interesting to note in figure 5.14 is that the model predicts gesture 4 as gesture 6 more than once. This could be an indication that these images show the most similar DFS profiles. The figure on the right confirms the solid performance of the CNN on the MNIST dataset. Where for most classes the scores are above 0.96, even the outliers for accuracy and recall still show an accuracy and recall of above 91% for the worst class.

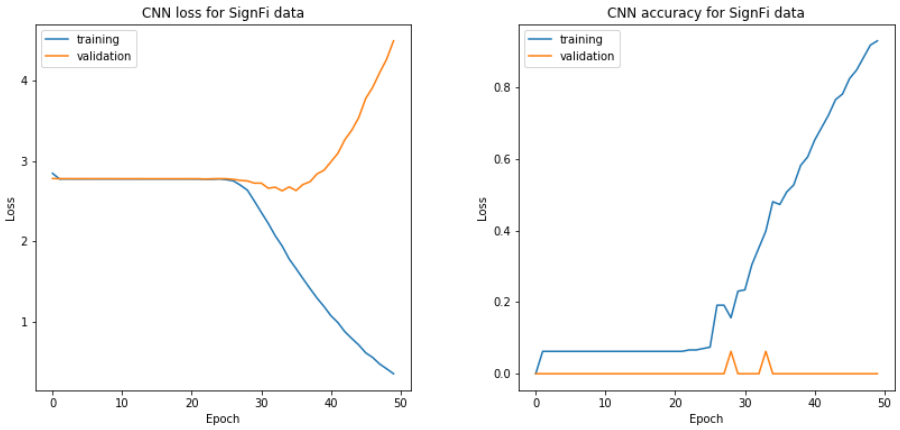

Figure 5.11: Loss and accuracy for CNN using SignFi.

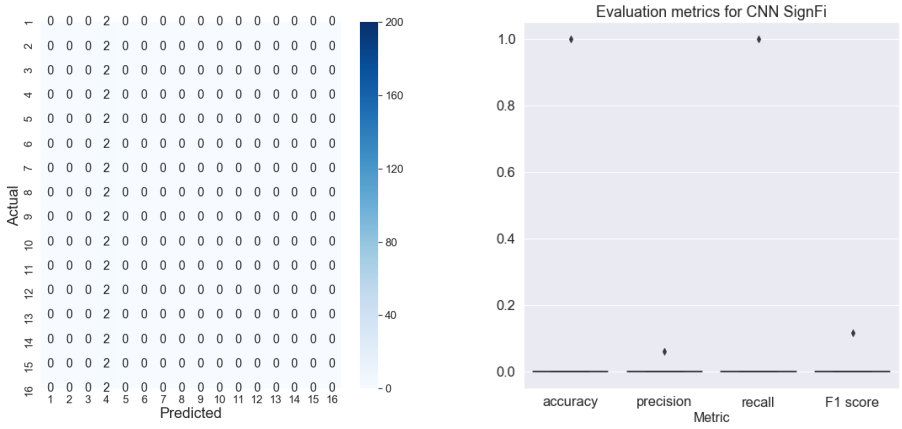

Figure 5.12: Confusion matrix and evaluation metrics for CNN using SignFi.

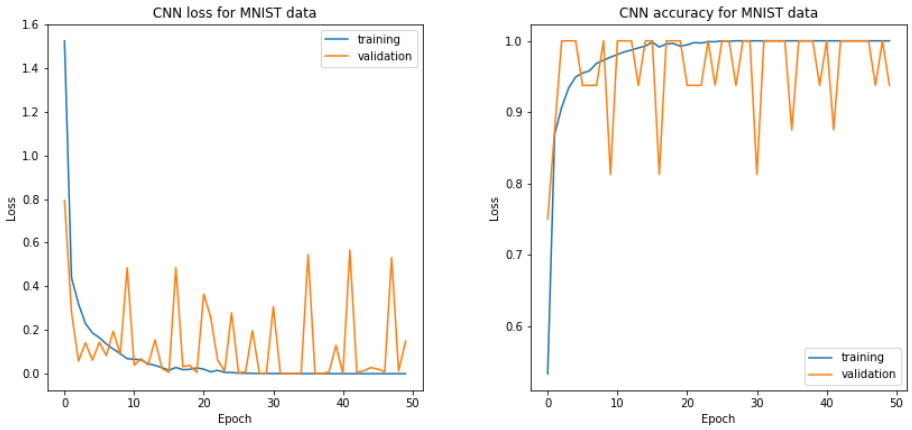

Figure 5.13: Loss and accuracy for CNN using MNIST.

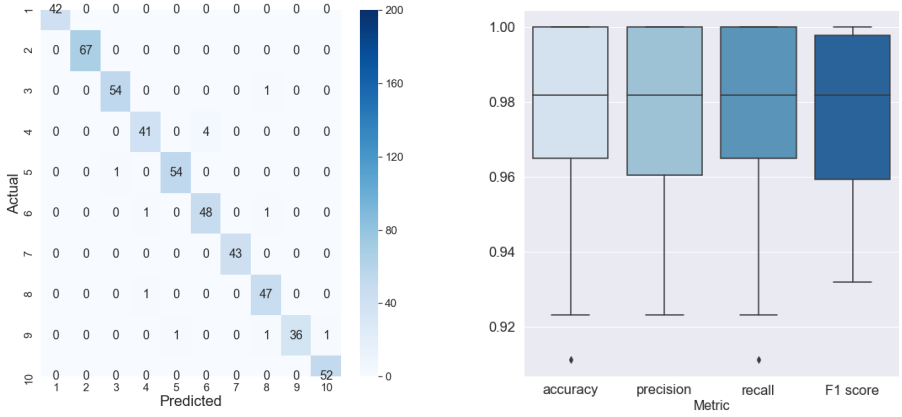

Figure 5.14: Confusion matrix and evaluation metrics for CNN using MNIST.

5.4 Discussion

This section will discuss some limitations of the implementation and the experiments and how these relate to the results.

Firstly, as mentioned before, the sampling for the MNIST training, testing and validation set is not stratified. For future research it is recommended to use stratification for all sampling. Since the subset used for the MNIST classifications is still quite large, it is possible to evaluate the models. However, the imbalance in classes influences the overall accuracy and the training process itself. It is crucial to emphasize once more that the input only contains the first 16 motions from the SignFi dataset in order to ensure reproducibility. This decision was made in order to conduct faster experiments. Furthermore, if the models cannot classify 16 gestures, they will be unable to classify 276 gestures.

In terms of the models, only few options out of many available options have been tested. It is important to realize that there are many more component settings. Not all

spike encoding methods have been evaluated and not all LIF neuron models have been used in the experiments. In addition, there are many more variables that influence the feasibility of spiking neural networks in WiFi CSI sensing. In this case only one model structure has been used with only one optimizer and only one loss function. No grid search is performed or any other standard methods for hyperparameter tuning.

With regard to for example 5.5, it is also worth it to explore different settings for the learning rate in future research. The decision was made to stick to default choices as much as possible, since these are likely to have solid performance in different settings. Think for example of using the Adam optimizer and a 0.001 learning rate. In terms of the learning rate, it would be possible to implement an adaptive learning rate. This would not guarantee better results but is worth exploring.

Both the spiking neural networks and the convolutional neural networks show very different performances for the two different datasets. Whereas different SNN models and the CNN achieve good results on MNIST, model performance does not improve significantly when trained on the SignFi dataset. In addition, some of the results can be caused by coincidence such as the result fro the SNN with Synaptic LIF where the accuracy is much better for the test set than for the train or validation set. The result of such different performances for different datasets is surprising but nevertheless interesting. To understand the cause of the differences, a thorough understanding of why the SNN performs well on the MNIST dataset and why the SNN does not continue to learn on the SignFi dataset is required. A possible answer can be found by understanding the differences between the two datasets.

One very clear difference is the size of the data. Even though a subset is taken from the MNIST dataset, there are still many more images from the model to learn from. Next to this, the sizes of the images themselves are also different, since the MNIST images contain less data points. As the same model structure is used for every experiment on SignFi and MNIST, the model itself does not play a major role in the contrasting results for this particular thesis even though it plays an important role in the overall performance.

There are two results that could be used as a starting point for further research. Firstly, the SNN and CNN tend to predict labels for the SignFi data to be in the same class. The class that is favoured differs per model. To understand if the number of images contributes to the nature of the models, it would be interesting to see what happens if only 20 images are used from each class of the MNIST data. To make an even better comparison, this could be supplemented with using only 10 gestures from the SignFi data.

Secondly, understanding why the models fail to generalize on the SignFi data would contribute to gainin more insights. For example, the convolutional neural network used, performs well on the training set, but is not able to deliver similar results for the validation and test data.

Another possible method that could be used to gain a better understanding of the differences in the data is to look at the probability distributions of the data. Since the data points in the input tensor are viewed as probabilities this could have a huge effect on the spike generation and therefore also the model.

The experiments with different inputs show that in this case for rate coding the opti-

mal setting is 50 steps. Increasing the number of time steps only leads to more complexity but not to a better model since it tends to overfit. For 50 epochs, the best setting for latency coding is the default setting of 0.01. The differences are small and it is interesting to note that the model does learn under all tested thresholds.

The Synaptic LIF neuron results in better model performance on the SignFi data than the Leaky LIF neuron but it would be needed to introduce a longer training process. For the MNIST dataset the Leaky LIF neuron shows better performance, even though both models show good results.

It is interesting to observe that for the third research question the Spiking Neural Network does learn from the training data. Nevertheless it fails to classify gestures in the validation and test set and is overfitting. CNN does outperform the SNN on the MNIST dataset for the current settings. Lastly, there are several occasions where the loss function is still decreasing for both the training and validation data. In order to properly compare the models, it is needed to run the training processes for more than 50 epochs.

Chapter 6

Conclusion

Where the initial expectation may have been that Spiking Neural Networks would be able to classify different gestures from the WiFi CSI data in SignFi and MNIST. The results have shown that this is not true for the used settings. The Spiking Neural Networks used perform very well for the MNIST dataset and achieve high performance overall. As for the SignFi dataset the Spiking Neural Networks either do not learn from the training data or only learn on the training data but fail to predict the gestures for the validation and test data. For the first research question we can conclude that the used model that takes rate spikes as input performs better when trained on 30 epochs and that the performance improves faster as for the model that takes the time-to-first spike as input. For the second research question it is shown that both the Leaky LIF neuron and the Synaptic LIF neuron can be used to classify images, at least for the MNIST data. Where a Convolutional Neural Network is able to classify the MNIST gestures well, while trained on 50 epochs, it also has trouble predicting the SignFi gestures.

Based on the results it is not possible to say that spiking neural networks can be used to classify all kinds of data. One conclusion is that spiking neural networks are suitable for WiFi CSI gesture recognition using the MNIST dataset and the settings used in this thesis. Furthermore, more research is necessary to understand under what circumstances spiking neural networks can and cannot learn. Before spiking neural networks can be used to address the domain shift problem, a more thorough understanding is needed of the effect of the data on the classification performance, a possible tendency to predict instances to be part of the same class and the influence of the amount of data that is used to train the model.

Bibliography

[1] *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way — by Sumit Saha — Towards Data Science*. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[2] *All about neural networks: Here's everything you need to know in 2022 - ai.nl*. URL: https://www.ai.nl/knowledge-base/neural-network/.

[3] Sheheryar Arshad et al. "Wi-chase: A WiFi based human activity recognition system for sensorless environments". In: *18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017 - Conference*. Institute of Electrical and Electronics Engineers Inc., July 2017. ISBN: 9781538627228. DOI: 10.1109/WoWMoM.2017.7974315.

[4] Daniel Auge et al. *A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks*. Dec. 2021. DOI: 10.1007/s11063-021-10562-2.

[5] Prachi Bagave and Ir DV Le Viet Duc. *Unobtrusive sensing using Wi-Fi signals*. Tech. rep. 2018.

[6] Adarsha Balaji et al. "Run-time Mapping of Spiking Neural Networks to Neuromorphic Hardware". In: (June 2020). URL: http://arxiv.org/abs/2006.06777.

[7] Jose M Benita et al. "Synaptic depression and slow oscillatory activity in a biophysical network model of the cerebral cortex". In: (2012). DOI: 10.3389/fncom.2012.00064. URL: www.frontiersin.org.

[8] *Binary Image classifier CNN using TensorFlow — by Sai Balaji — Techiepedia — Medium*. URL: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697.

[9] *Brain Basics: Know Your Brain — National Institute of Neurological Disorders and Stroke*. URL: https://www.ninds.nih.gov/health-information/public-education/brain-basics/brain-basics-know-your-brain.

[10] Jeroen Klein Brinke and Nirvana Meratnia. "Scaling activity recognition using channel state information through convolutional neural networks and transfer learning". In: *AIChallengeIoT 2019 - Proceedings of the 2019 International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*. Association for Computing Machinery, Inc, Nov. 2019, pp. 56–62. ISBN: 9781450370134. DOI: 10.1145/3363347.3363362.

[11] Maurice J Chacron et al. "Communicated by Paul Bressloff Interspike Interval Correlations, Memory, Adaptation, and Refractoriness in a Leaky Integrate-and-Fire Model with Threshold Fatigue". In: *Neural Computation* 15 (2003), pp. 253–278. URL: http://direct.mit.edu/neco/article-pdf/15/2/253/815471/089976603762552915.pdf.

[12] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. *Torch7: A Matlab-like Environment for Machine Learning*. Tech. rep. URL: http://numpy.scipy.org..

[13] Oscar Day and Taghi M. Khoshgoftaar. "A survey on heterogeneous transfer learning". In: *Journal of Big Data* 4.1 (Dec. 2017). ISSN: 21961115. DOI: 10.1186/s40537-017-0089-0.

[14] Nicki Skafte Detlefsen et al. "TorchMetrics-Measuring Reproducibility in PyTorch". In: (2022). DOI: 10.21105/joss.04101. URL: http://arxiv.org/abs/1907.11692.

[15] Peter U. Diehl and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity". In: *Frontiers in Computational Neuroscience* 9.AUGUST (Aug. 2015). ISSN: 16625188. DOI: 10.3389/fncom.2015.00099.

[16] Jason K Eshraghian et al. "TRAINING SPIKING NEURAL NETWORKS USING LESSONS FROM DEEP LEARNING". In: ().

[17] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: (Sept. 2021). URL: http://arxiv.org/abs/2109.12894.

[18] Eva Webster. *What is MIMO (multiple input, multiple output)?* URL: https://www.techtarget.com/searchmobilecomputing/definition/MIMO.

[19] Wulfram Gerstner and Werner M. Kistler. "Detailed neuron models". In: *Spiking Neuron Models*. Cambridge University Press, June 2012, pp. 31–68. DOI: 10.1017/cbo9780511815706.003.

[20] *GitHub - aladdinpersson/Machine-Learning-Collection: A resource for learning about Machine learning & Deep Learning*. URL: https://github.com/aladdinpersson/Machine-Learning-Collection.

[21] Wenzhe Guo et al. "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems". In: *Frontiers in Neuroscience* 15 (Mar. 2021). ISSN: 1662453X. DOI: 10.3389/fnins.2021.638474.

[22] Guo li Taiwan da xue et al. *ISLPED 2017 : IEEE/ACM International Symposium on Low Power Electronics and Design : July 24-26, 2017 @ Taipei, Taiwan.* ISBN: 9781509060238.

[23] Chunmei Han et al. "WiFall: Device-free Fall Detection by Wireless Networks". In: ().

[24] Charles R. Harris et al. *Array programming with NumPy*. Sept. 2020. DOI: 10.1038/s41586-020-2649-2.

[25] Wenfeng He et al. "WiG: WiFi-based gesture recognition system". In: *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. Vol. 2015-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2015. ISBN: 9781479999644. DOI: 10.1109/ICCCN.2015.7288485.

[26] *How WiFi Works — HowStuffWorks*. URL: https://computer.howstuffworks.com/wireless-network.htm.

[27] Xin Huang et al. "Deep Learning Based Solar Flare Forecasting Model. I. Results for Line-of-sight Magnetograms". In: *The Astrophysical Journal* 856.1 (Mar. 2018), p. 7. ISSN: 15384357. DOI: 10.3847/1538-4357/aaae00.

[28] Somayeh Hussaini, Michael Milford, and Tobias Fischer. "Spiking Neural Networks for Visual Place Recognition Via Weighted Neuronal Assignments". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 4094–4101. ISSN: 23773766. DOI: 10.1109/LRA.2022.3149030.

[29] Zhiping Jiang et al. "Eliminating the Barriers: Demystifying Wi-Fi Baseband Design and Introducing the PicoScenes Wi-Fi Sensing Platform". In: (Oct. 2020). URL: http://arxiv.org/abs/2010.10233.

[30] Chankyu Lee et al. "Training deep spiking convolutional Neural Networks with STDP-based unsupervised pre-training followed by supervised fine-tuning". In: *Frontiers in Neuroscience* 12.AUG (Aug. 2018). ISSN: 1662453X. DOI: 10.3389/fnins.2018.00435.

[31] Vijaysinh Lendave. *A Tutorial on Spiking Neural Networks for Beginners*. Nov. 2021.

[32] Jiguang Lv, Wu Yang, and Dapeng Man. "Device-free passive identity identification via WiFi signals". In: *Sensors (Switzerland)* 17.11 (Nov. 2017). ISSN: 14248220. DOI: 10.3390/s17112520.

[33] Yongsen Ma, Gang Zhou, and Shuangquan Wang. *WiFi sensing with channel state information: A survey*. June 2019. DOI: 10.1145/3310194.

[34] Yongsen Ma et al. "SignFi". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (Mar. 2018), pp. 1–21. DOI: 10.1145/3191755.

[35] Sébastien Marcel and Yann Rodriguez. "Torchvision the machine-vision package of Torch". In: (2010). URL: http://www.torch.ch.

[36] Pedro Melgarejo et al. "Leveraging directional antenna capabilities for fine-grained gesture recognition". In: *UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. Association for Computing Machinery, Inc, 2014, pp. 541–551. ISBN: 9781450329682. DOI: 10.1145/2632048.2632095.

[37] Gustavo B.M. Mello, Sofia Soares, and Joseph J. Paton. "A scalable population code for time in the striatum". In: *Current Biology* 25.9 (May 2015), pp. 1113–1122. ISSN: 18790445. DOI: 10.1016/J.CUB.2015.02.036.

[38] Sushmita Mitra and Tinku Acharya. "Gesture recognition: A survey". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 37.3 (May 2007), pp. 311–324. ISSN: 10946977. DOI: 10.1109/TSMCC.2007.893280.

[39] Kai Niu et al. “Understanding WiFi signal frequency features for position-independent gesture sensing Transactions on Mobile Computing IEEE TRANSACTIONS ON MOBILE COMPUTING 1 Understanding WiFi Signal Frequency Features for Position-Independent Gesture Sensing”. In: (2021), pp. 1–16. DOI: 10.1109/TMC.2021.3063135{\"{i}}. URL: https://youtu.be/o6IbReBig.

[40] Keiron O’Shea and Ryan Nash. “An Introduction to Convolutional Neural Networks”. In: (Nov. 2015). URL: http://arxiv.org/abs/1511.08458.

[41] C H J M Oerlemans. *The Effect of Data Preprocessing On the Performance of Few-shot Learning for Wi-Fi CSI-based Gesture Recognition*. Tech. rep.

[42] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: (2019).

[43] Fabian Pedregosa FABIANPEDREGOSA et al. “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://scikit-learn.sourceforge.net..

[44] Tomaso Poggio et al. “Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review”. In: *International Journal of Automation and Computing* 14.5 (Oct. 2017), pp. 503–519. ISSN: 17518520. DOI: 10.1007/S11633-017-1054-2. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[45] Nitin Rathi et al. *STDP Based Pruning of Connections and Weight Quantization in Spiking Neural Networks for Energy-Efficient Recognition*. Tech. rep.

[46] Monika Schak and Alexander Gepperth. “Gesture MNIST: A New Free-Hand Gesture Dataset”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13532 LNCS (2022), pp. 657–668. ISSN: 16113349. DOI: 10.1007/978-3-031-15937-4{_}55/FIGURES/6. URL: https://link.springer.com/chapter/10.1007/978-3-031-15937-4_55.

[47] Jiacheng Shang and Jie Wu. “A robust sign language recognition system with multiple Wi-Fi devices”. In: *MobiArch 2017 - Proceedings of the 2017 Workshop on Mobility in the Evolving Internet Architecture, Part of SIGCOMM 2017*. Association for Computing Machinery, Inc, Aug. 2017, pp. 19–24. ISBN: 9781450350594. DOI: 10.1145/3097620.3097624.

[48] Abdallah Sobehy, Eric Renault, and Paul Mühlethaler. “CSI-MIMO: K-nearest neighbor applied to indoor localization”. In: (). URL: https://hal.archives-ouvertes.fr/hal-02491175.

[49] Ioana Sporea and André Grüning. “Supervised Learning in Multilayer Spiking Neural Networks”. In: *Neural Computation* 25.2 (Feb. 2012), pp. 473–509. DOI: 10.1162/NECO{_}a{_}00396. URL: http://arxiv.org/abs/1202.2249%20http://dx.doi.org/10.1162/NECO_a_00396.

[50] Erik R Svensson. *A comparison between Feed-forward and Convolutional Neural Networks for classification of invoice documents*. Tech. rep.

[51] Sheng Tan and Jie Yang. "WiFinger: Leveraging commodity WiFi for fine-grained finger gesture recognition". In: *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. Vol. 05-08-July-2016. Association for Computing Machinery, July 2016, pp. 201–210. ISBN: 9781450341844. DOI: 10.1145/2942358.2942393.

[52] Amirhossein Tavanaei et al. "Deep Learning in Spiking Neural Networks". In: (Apr. 2018). DOI: 10.1016/j.neunet.2018.12.002. URL: http://arxiv.org/abs/1804.08150%20http://dx.doi.org/10.1016/j.neunet.2018.12.002.

[53] The pandas development team. "pandas-dev/pandas: Pandas". In: (Nov. 2022). DOI: 10.5281/ZENODO.7344967. URL: https://doi.org/10.5281/zenodo.7344967#.Y8B9TLtB_Qk.mendeley.

[54] Hasmath Farhana Thariq Ahmed, Hafisoh Ahmad, and Aravind C.V. "Device free human gesture recognition using Wi-Fi CSI: A survey". In: *Engineering Applications of Artificial Intelligence* 87 (Jan. 2020). ISSN: 09521976. DOI: 10.1016/j.engappai.2019.103281.

[55] Johannes C Thiele, Olivier Bichler, and Antoine Dupret. "SpikeGrad: An ANN-equivalent Computation Model for Implementing Backpropagation with Spikes". In: ().

[56] *Understanding RSSI*. 2022.

[57] Raghav H. Venkatnarayan, Griffin Page, and Muhammad Shahzad. "Multi-user gesture recognition using WiFi". In: *MobiSys 2018 - Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, Inc, June 2018, pp. 401–413. ISBN: 9781450357203. DOI: 10.1145/3210240.3210335.

[58] Aditya Virmani and Muhammad Shahzad. "Position and orientation agnostic gesture recognition using WiFi". In: *MobiSys 2017 - Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, Inc, June 2017, pp. 252–264. ISBN: 9781450349284. DOI: 10.1145/3081333.3081340.

[59] Mei Wang and Weihong Deng. "Deep visual domain adaptation: A survey". In: *Neurocomputing* 312 (Oct. 2018), pp. 135–153. ISSN: 18728286. DOI: 10.1016/j.neucom.2018.05.083.

[60] William G. Wong. *Wi-Fi Sensing: The Next Big Wireless Movement*. Feb. 2022.

[61] Dan Wu et al. "WiFi CSI-based device-free sensing: from Fresnel zone model to CSI-ratio model". In: *CCF Transactions on Pervasive Computing and Interaction* 4.1 (Mar. 2022), pp. 88–102. ISSN: 25245228. DOI: 10.1007/s42486-021-00077-z.

[62] Zheng Yang et al. *Smart Wireless Sensing*. Springer Singapore, 2021. DOI: 10.1007/978-981-16-5658-3. URL: https://link.springer.com/chapter/10.1007/978-981-16-5658-3_2.

[63] Yuqing Yin et al. “Towards fully domain-independent gesture recognition using COTS WiFi device”. In: *Electronics Letters* 57.5 (Mar. 2021), pp. 232–234. ISSN: 1350911X. DOI: 10.1049/ell2.12097.

[64] Daqing Zhang, Hao Wang, and Dan Wu. *Toward Centimeter-Scale Human Activity Sensing with Wi-Fi Signals*. Tech. rep. URL: www.computer.org/computer-multimedia.

[65] Daqing Zhang et al. “Practical Issues and Challenges in CSI-based Integrated Sensing and Communication”. In: (Mar. 2022). URL: http://arxiv.org/abs/2204.03535.

[66] Yue Zheng et al. “Zero-effort cross-domain gesture recognition with Wi-Fi”. In: *MobiSys 2019 - Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, Inc, June 2019, pp. 313–325. ISBN: 9781450366618. DOI: 10.1145/3307334.3326081.

[67] Qizhen Zhou et al. “A device-free number gesture recognition approach based on deep learning”. In: *Proceedings - 12th International Conference on Computational Intelligence and Security, CIS 2016*. Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 57–63. ISBN: 9781509048403. DOI: 10.1109/CIS.2016.21.

[68] Augustinas Zinys, Bram van Berlo, and Nirvana Meratnia. “A domain-independent generative adversarial network for activity recognition using wifi csi data”. In: *Sensors* 21.23 (Dec. 2021). ISSN: 14248220. DOI: 10.3390/s21237852.