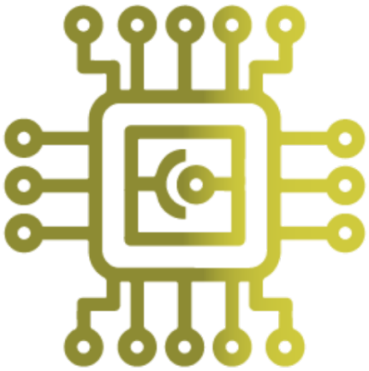
Ref. Ares(2025)9370323 - 31/10/2025



CONVOLVE

Seamless design of smart edge processors

GRANT AGREEMENT NUMBER: 101070374

Deliverable D4.4

Report on progress in SNNs and learning algorithms

Disclaimer
This project has received funding from the European Union's Horizon 2021 research and innovation programme under grant agreement No 101070374. This document has been prepared for the European Commission, however, it reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Title of the deliverable	Report on progress in SNNs and learning algorithms
WP contributing to the deliverable	WP4
Task contributing to the deliverable	T4.3, T4.4
Dissemination level	PU – Public
Due submission date	2025-10-31
Actual submission date	2025-10-31
Author(s)	Mattias Nilsson (FMI) Friedemann Zenke (FMI) Benjamin Cramer (BOS) Shreya Kshirasagar (BOS) Andre Guntoro (BOS) Simon Davidson (MAN) James Garside (MAN)
Internal reviewers	Sander Stuijk (TUE)

Document Version	Date	Change
V0.1	2025-04-09	Table of contents and main document structure
V1.0	2025-10-17	First integrated version
V1.1	2025-10-31	Revision following internal review

Table of Contents

Table of Contents 3
Deliverable summary 5
1. Introduction 5
1.1. Background and motivation 6
1.1.1. Spiking neural networks 6
1.1.2. Online temporal learning 6
1.2. Objectives of deliverable 6
2. Methodology for ANN to SNN conversion (MAN) 7
3. Novel compute mechanisms for SNNs (MAN) 8
3.1. Motivation and outline of mechanisms 8
3.2. Exploiting spike time in SNN computation 8
3.3. Detecting patterns across multiple temporal scales 9
3.4. Exploiting sparsity in SNN computation 10
3.5. Developed framework 10
3.6. Impact of parameters on integration of dendritic input 11
3.7. Training methodology 12
3.8. Evaluation methodology using applications from WP1 12
3.9. Current status and ongoing work 14
4. SNN demonstrators for target applications (BOS, FMI) 15
4.1. Acoustic scene analysis (BOS) 15
4.1.1. Auditory anomaly detection 15
4.1.2. Sound source tracking 16
4.2. Speech quality prediction (FMI) 18
4.2.1. Outline of application 18
4.2.2. Model 18
4.2.3. Handling temporal sparsity 19
4.2.4. Demonstrator 19
4.2.5. Results 20
4.2.6. Challenges and limitations 21
5. Online learning strategies (FMI) 21
5.1. Online learning algorithm 21
5.1.1. Overview 21
5.1.2. Preliminary results and future work 22
5.2. Handling temporal sparsity 23

6. Conclusions and future work 23
6.1. Summary of key achievements 23
6.2. Remaining challenges and open questions 23
7. References 24
8. Annexes 30
8.1. Links to code 30
8.1.1. Speech quality prediction with SNNs 30

Deliverable summary

This report presents the results from tasks T4.3 and T4.4 within WP4 of the CONVOLVE project.

Within T4.3, spiking neural network (SNN) models were developed for three auditory target applications: auditory anomaly detection, sound source tracking, and speech quality prediction (SQP). BOS developed a recurrent SNN for auditory anomaly detection and a convolutional SNN for sound source tracking, both of which perform comparatively to baseline ANNs while using fewer or an equal number of parameters and having potential for efficient spike-based hardware implementations. For SQP, the FMI developed a convolutional SNN with comparable performance to existing lightweight ANNs based on CNN or LSTM, while reducing estimated FLOPs during inference by 2.5-5x for sparsity-aware hardware. FPGA-based hardware implementations of the SNNs for anomaly detection and SQP are under development by MAN. The SNN for sound source tracking is being investigated for hardware implementation by TUE. Details about the SNN hardware accelerators are found in Deliverable D2.4.

Within T4.4, the FMI are developing a forward-in-time learning algorithm for SNNs in which simplifying approximations are introduced to the dynamics of neurons and synaptic eligibility traces to reduce their computational cost on digital hardware. Preliminary results indicate that the approximations do not cause much performance decrease for the basic case of a feedforward SNN with a single hidden layer. Future work will investigate their application in recurrent and deep SNNs and quantitatively estimate efficiency impact.

1. Introduction

This document, D4.4 "Report on progress in SNNs and learning algorithms", is a deliverable of WP4 "Algorithmic principles for ultra-low power neural network processing" related to the following tasks:

- T4.3: "Develop spiking networks for target applications" under the task lead of MAN,
- T4.4: "Learning strategies" under the task lead of FMI.

The document covers the following topics:

- Sec. 2: Methodology for ANN to SNN conversion,
- Sec. 3: Novel compute mechanisms for SNNs,
- Sec. 4: SNN demonstrators for target applications,
    - Auditory anomaly detection,
    - Sound source tracking,
    - Speech quality prediction,
- Sec. 5: Online learning strategies.

1.1. Background and motivation

1.1.1. Spiking neural networks

Spiking neural networks (SNNs) (Maass, 1997; Eshraghian et al., 2023) have potential for efficient streaming inference on dedicated, sparsity-aware ("neuromorphic") hardware (Kudithipudi et al., 2025) by virtue of their event-driven processing and sparse, binary activations (spikes). While SNNs have, for instance, shown promise for low-power audio processing in tasks like ambient audio classification (Bos & Muir, 2023), keyword spotting (Bos & Muir, 2024), and speech enhancement (Sun & Bohté, 2024; Hao et al., 2025; Yan et al., 2025), there is still a need to develop SNNs with demonstrated competitive performance and efficiency benefits on more real-world tasks.

One such potential task is speech quality prediction (SQP), which is part of GNA's use-case on speech enhancement within CONVOLVE WP1. Previous work on models for efficient SQP has relied on either 1D temporal or 2D time–frequency convolutional architectures (Z. Li & Li, 2023; Cumlin et al., 2024). These models, while being lightweight in terms of number of parameters, process entire multi-second temporal windows for each prediction update. This sliding window pattern gives rise to inherent computational redundancy in streaming scenarios, which limits the efficiency for real-time applications. SNNs, on the other hand, process sequences causally, one step at a time, in a way that is naturally suited for streaming inference over indefinite sequence lengths.

1.1.2. Online temporal learning

For deep learning with temporal data, such as audio, the two main algorithms for exact gradient computation are (i) backpropagation through time (BPTT), which is non-local in time, and (ii) real-time recurrent learning (RTRL), which has very high compute-and-memory scaling. Therefore, to enable forward-in-time learning suitable for online deployment, some approximation of RTRL is required.

One method for reducing the costs of RTRL is to use neural network architectures with diagonal recurrence . This property reduces the compute-and-memory scaling of RTRL and is inherent by default in feedforward SNNs as well as diagonal or block-diagonal state-space models. Still, different design choices are involved in formulating such an RTRL algorithm for SNNs (Eshraghian et al., 2023), affecting computational complexity and performance. One open question is what fidelity is needed in modeling the continuous states that are part of both the SNN and the learning algorithm, as the associated frequent state updates are computationally costly for digital hardware implementations.

1.2. Objectives of deliverable

Based on the roadmap laid out in D4.1, T4.3 set out to develop SNN models and investigate their possible benefits over ANNs in certain target applications. In particular, the SNN development targeted applications with streaming or event-driven input, where efficiency benefits over ANNs may be expected. These SNNs aimed at >3x savings in computational cost against ANNs for the same tasks.

T4.4 addresses energy-and-memory efficient learning algorithms suitable for online temporal learning in edge chips. This involves investigating algorithms that avoid the temporal non-locality of BPTT as well as the unfeasible compute-and-memory scaling of naïve RTRL.

2. Methodology for ANN to SNN conversion (MAN)

The recent surge in applications of ANNs has taken them from the academic realm into the mainstream, with ANN-powered products such as ChatGPT, Dall-E and Midjourney suddenly finding their way into the hands of every consumer. Spiking networks, in comparison, remain of only academic interest. One way for SNNs to catch up is to convert pre-trained ANNs and this approach has been pursued in academia (Rueckauer & Liu, 2018).

In ANN-to-SNN conversion, each neuron in a pre-trained ANN is mapped to an equivalent spiking neuron. Connections are likewise mapped one-to-one. The output activation of the ANN neuron then becomes a series of spikes in the SNN, with higher activation corresponding to higher spike rate.

Our own work (completed prior to CONVOLVE) had convinced us that using high spike rates was not an energy efficient way of using spiking networks (Davidson & Furber, 2021), but the doctoral work of a PhD student in our group at Manchester, Chen Li, lent support to a more refined version of ANN to SNN conversion that might still be a worthwhile approach (C. Li, 2022). Thus, our early focus on CONVOLVE was to establish the utility and universality of the Chen Li method. This method called for very low spike rates (at most three spikes per transmitted value).

Furthermore, Chen uses the same extreme quantization during the training of the ANN (integer output activations in the range of zero to three), so that the translation process to an SNN would be as seamless as possible. One final innovation proposed by Chen was the *negative spike*, intended to undo the action of a speculatively issued spike that was subsequently found to be unwanted. This allowed spikes to be generated when only part of the neuron's intended input had arrived, speeding processing. In many cases this early spike would still have been generated once all of the intended input had arrived (so no harm was done), but in some cases it would be found that later inhibitory input would have prevented the spike from being generated. The issuing of a negative spike would reverse the actions of the earlier spike, restoring correct operation.

In his thesis, Chen presented results suggesting that the loss of performance accuracy with his changes would be negligible. He simulated his networks using his own C code using an MNIST recognition network. However, our own investigations running on the SpiNNaker platform were less successful. We were unable to implement the negative spike concept on SpiNNaker without significant changes to the software model, which we lacked the expertise to implement. The lack of negative spikes had an impact on performance. Furthermore, Chen's example network did not include some of the more complex layer types, such as max pooling, that are needed for general purpose computing with spiking networks. We found that the three-spike limit caused a significant loss of performance with the max pooling function, leading us to conclude that such spike resolution limitations are problematic for general neuromorphic computation.

We thus decided after six months that the Chen methods, and by extension the whole approach of converting an ANN into a spike-rate coded SNN, were not a fruitful methodology and that designing directly with the spikes with time-sensitive computation was more likely to yield solutions to compete with the dominant ANNs in the market today.

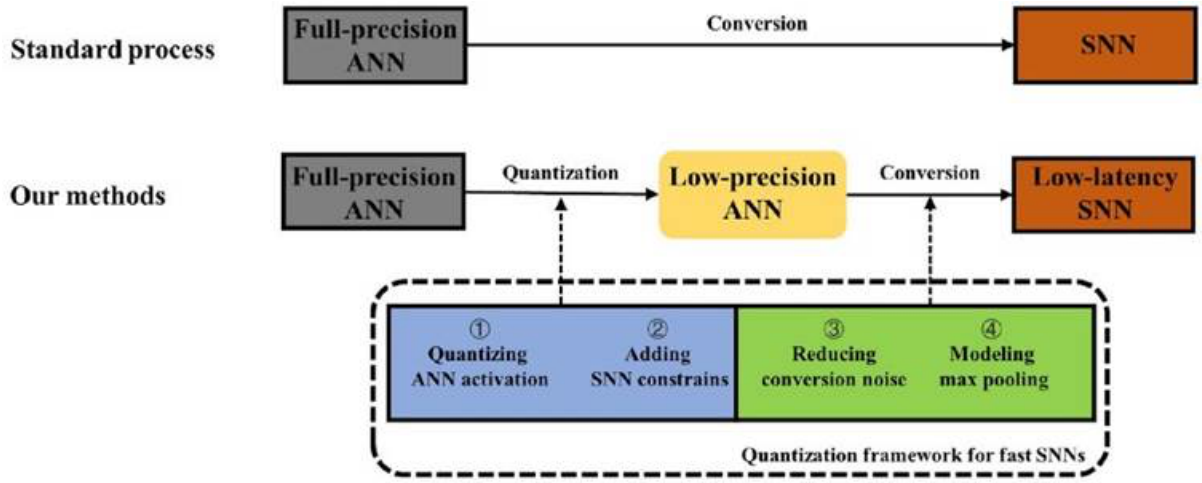


FIGURE 2.1: FLOW CHART FOR TWO METHODOLOGIES TO CONVERT AN ANN INTO AN SNN. THE LOWER METHODOLOGY SHOWS CHEN LI'S METHOD, WHICH USES VERY SPARSE SPIKE REPRESENTATION AND QUANTIZES THE ANN DURING TRAINING TO MATCH THIS SPARSITY FROM THE BEGINNING OF TRAINING.

3. Novel compute mechanisms for SNNs (MAN)

3.1. Motivation and outline of mechanisms

Conventional neural networks (ANNs) differ from spiking neural networks (SNNs) in several ways, most obviously in the form of the data that passes from one neuron to another during computation. In an ANN, each neuron sends a multi-valued activation value on each tick of the clock. In an SNN, neurons communicate by sending single event packets known as spikes. Individually, a spike conveys much less information than the activations produced by an ANN.

However, an intrinsic property of spiking neural networks is that the time dimension is fundamental to their operation. Each neuron carries state in the form of its membrane potential, which is boosted by incoming excitation from other neurons while being decayed with each time step of simulation. This evolving state renders the spiking neuron a more complex computing unit than the simpler ANN neuron, which typically recomputes its potential from zero on each simulation timer tick. The time of production of a spike can convey additional—exploitable—information. However, this information is often ignored in spiking networks, which often use multi-spike rate coding to communicate between neurons, throwing away timing information in the process.

3.2. Exploiting spike time in SNN computation

Since the timing of spikes is exploitable information, the next question is how to utilize spike time for computation. A simple neuron calculates and maintains a weighted sum of their inputs in the form of its membrane potential, with an output spike generated if the potential exceeds some threshold. The potential is reset when this occurs. Since the potential decays over time,

co-incident arriving spikes are more likely to trigger an output event than incoming spikes that are well separated in time. Thus, the basic operation of the neuron is to act as an approximate co-incidence detector with weighting. This opens up the possibility of using inter-neuronal delays as a means of shaping the interactions between neurons, a capability that exists in only a rudimentary form with ANNs, where activity proceeds from layer to layer one timer tick at a time during simulation.

The concept of using inter-neuronal delays to detect a particular pattern of spike times from a population of neurons is not new. Twenty years ago, Izhikevich proposed a mechanism, *polychronization* (Izhikevich, 2006) to insert delays between neurons to bring a set of incoming spikes into alignment, thus allowing a specific pattern of spike times to be detected by the receiving neuron. However, this concept has not been widely adopted, and so there is scope to explore uncharted research territory. In particular, his formulation assumed one neuron being deployed to detect one specific pattern only. A more distributed representation would be a more efficient use of compute resources, with a single neuron responding to multiple patterns (and aliases due to combinations of valid patterns). This is discussed later.

3.3. Detecting patterns across multiple temporal scales

The concept of detecting co-incident spikes does raise the question of how to define *co-incident*. The membrane potential retains a trace of each arriving spike by summing all of the individual contributions, while the membrane potential decay acts to forget the contribution of previous spikes over time. Neurons assigned a small time constant will rapidly erase the contribution of an arriving spike so that for co-incidence to be detected, incoming spikes must arrive close together. Such a neuron will therefore be sensitive to the spike times but will only be capable of detecting features and patterns that occur closely grouped in time, limiting the type of patterns they can detect.

In contrast, neurons assigned a large time constant will retain history for longer, allowing spikes to be treated as co-incident even with large inter-spike intervals. Here, the detected pattern can occur over a long time scale, but the neuron is less sensitive to the timing of individual spikes and would be less able to respond differently to similar but distinct patterns with small changes in spike times.

Since both spike-time sensitivity and the ability to combine information over long time scales are desirable but competing qualities for these simple neuron models, one approach is to provide a heterogeneous set of compute units representing both ends of this spectrum and many points in between.

Possible ways to do this that were investigated:

1. Heterogeneous membrane time constants: Use simple neurons but assign each one a different membrane decay time constant. This creates a heterogeneous population with each neuron responding differently to variations of the same stimuli. Collectively, the population could have the capability to recognize features at both fine and coarse granularity.
2. Dendritic branches: Replace the simple passive dendritic tree with more complex morphology, mimicking nature more closely (Branco & Häusser, 2010). Individual branches act as local integrators, mini-neurons with their own potential and decay

constant: the decay constant is short for branches and long for the main cell. This allows branches to be time sensitive to individual features and less time-sensitive to their combination to detect patterns of multiple features. Features detected in a branch would only be signaled to the parent neuron.

3. **Alternating layers:** Mimic the dendritic trees using only a combination of simple neurons. Create alternating layers of basic neurons with short time membrane decay time constants and layers comprising neurons with long time constants. Here, features detected in the neurons with short time constants could be distributed to all neurons in the next layer, with more opportunity to combine features in a useful way, but with more work to distribute and combine the outputs of all neurons.

The simplest and most general of these is case (1), a population of neurons with distinct membrane decay time constants.

3.4. Exploiting sparsity in SNN computation

High sparsity (the pruning of a high percentage of the potential connections between two populations of neurons) directly contributes to energy saving since the energy cost of a neural network is typically dominated by the fetching and processing of connections in response to an incoming spike. Without careful design, however, high sparsity also impacts performance through a drop in accuracy and ability to generalize (Cheng et al., 2024; Farina et al., 2024).

Typical approaches to training sparse networks prune connections whose loss will minimally impact the performance of the network. Thus, the pruning occurs as a function of the training dataset. An alternative approach, and one that seemed more appropriate for CONVOLVE, was to start with a set of pre-pruned connections (to a specified degree of sparsity) and to force the network to learn using the remaining extant connections. This approach should enable the network to continue to learn using the same learning rule while the network is deployed in the field. This addresses the CONVOLVE project goal of **supporting continual learning**.

The fact that networks with sparse connectivity can still yield performance close to that of fully-connected networks suggests that the missing connections were passing on unhelpful information that was not needed for computation. Therefore, starting with a pruned network and learning to use the connections that are present can be interpreted as finding the optimal encoding at each layer that gets needed information to where it is most useful while discarding unnecessary information.

3.5. Developed framework

Given the design constants discussed in earlier sections, the framework that give the most promising results with the least neuronal complexity is as follows.

- **Neurons** are simple leaky-integrate-and-fire (LIF) models, with a trainable threshold (as a conscience mechanism) but with a membrane time constant selected from a range of valid values **(heterogeneous membrane decay time constants)**.
- **Dendritic inputs** are passive (i.e. no complex non-linearities in the dendritic tree nor interactions between processed inputs until they reach the neuron soma for summation with the neuron potential.

- **Dendritic delays** of up to 16 timesteps. Thus, a neuron is sensitive to precise spike timings over a window into the past of 16 timesteps.
- **Synaptic shaping** is implemented as a delta function so that an arriving spike adds a pulse of charge—equal to the synaptic weight—to the total potential after a delay defined for that dendrite (see above).
- **Sparse connectivity**. We have aimed for 10% sparsity or less, but the results so far have been mixed at this level of pruning.

3.6. Impact of parameters on integration of dendritic input

The figure below shows an example of a model neuron with a set of dendrites. The length of each dendrite corresponds to the delay it introduces to spikes that arrive at its synapse. The set of circles on the left represents the arriving signals (called axons) from an earlier layer. Since sparse connectivity is employed between layers, only a fraction of the incoming axons actually synapse onto dendrites of this model neuron. For illustrative purposes, the top four dendrites all have short delays, while the bottom four dendrites all have long delays. In reality, delays are randomized at the inception of the network before training.

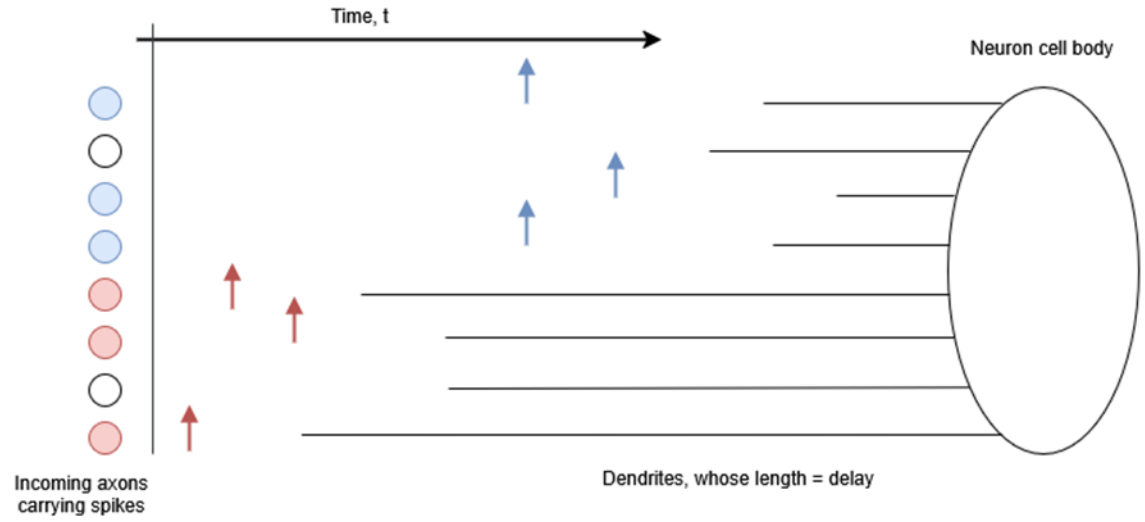


FIGURE 3.1: SINGLE NEURON SAMPLING FROM A POPULATION OF INCOMING AXONAL CONNECTIONS FROM THE PREVIOUS LAYER. THE LENGTH OF THE DENDRITES REPRESENTS THE DELAYS APPLIED TO EACH DENDRITE. DISTINCT FEATURES ARRIVE ON RED AND BLUE SETS OF AXONS. HOW SENSITIVE THE NEURON IS TO EACH FEATURE OR THEIR COMBINATION CAN BE TUNED USING THE MEMBRANE DECAY TIME CONSTANT AND THE NEURON THRESHOLD.

The figure shows two distinct sets of spikes, red and blue. The red set of spikes arrive first and represent one feature in the input image. They match the pattern of dendritic delays for those dendrites and so the spikes will arrive at the cell body close together, causing the neuron to fire, signaling the detection of the desired feature. The behavior in response to variations of this red spike pattern depends on the threshold and decay time constant of the neuron.

If the decay constant is small (i.e. rapid decay), even small deviations from the desired pattern can cause the neuron to fail to fire. Setting the threshold too high will also cause it to remain silent except for a very close or even perfect match against the desired pattern. In the short-time constant scenario, the same neuron may also be trained to match against an unrelated feature on the blue input axons. These spikes arrive later but are themselves close together in

time and are well matched by the dendritic delays so that they too can trigger the output neuron. The effect of the arriving red and blue spikes can occur close together with minimal interference since the short time constant causes the effect of the red spikes to dissipate before the blue spikes arrive. Overall, the neuron is sensitive to precise spike timing but is intolerant of deviations.

In an alternative version of this neuron with a long membrane decay time constant, the neuron can detect a wider range of possible red spike patterns with significant deviations from the exact spike timing. It can also retain knowledge of the red feature for longer before the arrival of the feature present on the blue inputs so that more temporal distribution patterns can be detected. This is important because it is wasteful to set up a set of neurons all tuned to make the spikes from a single pattern line up and make the neuron fire. With correct parameterization, a neuron could be able to support a set of independent feature detectors that may fire at different times but be able to detect combinations of these separate detectors, increasing the number of detected patterns a single neuron can discriminate.

However, the cost is that the neuron cannot discern fine distinctions between patterns that might have very different meanings. Thus, a range of different responses working together can provide a richer interpretation of the input stimulus.

3.7. Training methodology

In common with most spiking networks, a surrogate gradient learning rule was adopted, along with a variable threshold which falls slowly (over many training epochs) if the neuron is not firing enough and rises slowly if the neuron fires too much. This imposes sparsity on the activity of the network and acts as a conscience mechanism, forcing neurons to distribute the work of representing features in the input stream.

3.8. Evaluation methodology using applications from WP1

Experiments using the framework were implemented in Python using the PyTorch and snnTorch libraries.

Given that the way of representing and computing in this framework is significantly different from that used in the target applications in WP1, it is not possible to perform a one-to-one translation of the networks supplied in WP1 by the industrial partners. The only reasonable approach is to begin with the training data (input and desired outputs) and construct an equivalent network out of our new framework elements to perform the same task.

From the target applications on offer, the speech denoising network, NSNet2, from GNA seemed the most appropriate since it is of manageable size for simulation, used fully connected layers, and required spatiotemporal processing, which was an intuitive fit with the compute mindset of spiking neurons.

The satellite image processing model from Thales is based on convolutional networks, processes a series of static images and is large in size, making it awkward for rapid simulation. Thus, this network was not desirable for early experimentation.

We have not attempted to redesign partner applications that are spiking in nature, such as the siren networks from Bosch and the speech quality prediction network from FMI, since each of these incorporates their own unique experimental ideas in spiking networks, which would represent additional unique challenges for us that were specific to their existing implementation.

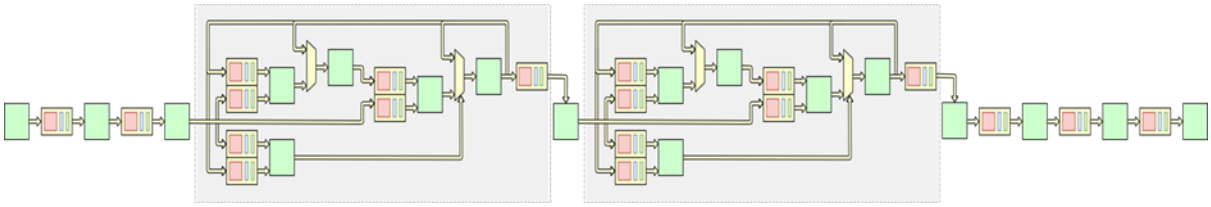


FIGURE 3.2: STRUCTURE OF THE NSNET2 NETWORK AS DESCRIBED BY GNA FOR THEIR SPEECH DENOISING APPLICATION. MULTI-COLORED BOXES ARE SINGLE NEURONAL LAYERS; GREEN BOXES ARE BUFFERING BETWEEN LAYERS. EACH GREY GROUPING REPRESENTS A GATED RECURRENT UNIT (GRU). LAYERS ARE FULLY CONNECTED INTERNALLY.

To avoid a complete redesign of each application, the approach taken was to use the same basic structure as the original model: identical numbers of layers and inter-layer connectivity. The size of each layer was also taken from the original model, except for input which had come from a spatiotemporal source, which was the case for the NSNet2 model from GNA. Such models have converted a time-based signal across C channels into a two-dimensional frequency and spatial stream that is cut up into chunks of fixed size F x T, where F is the number of frequency components and T is the number of time steps in the window. For our model, we take raw stream data from C channels and feed it into the model one spatial row at a time, removing the need for both a window and an FFT. In principle, this collapses the time dimension to a single element per channel at each time step and would, on its own, reducing the number of input units from FT in the original model to only C in the new model.

To make this work, all features that were being captured by the FT unit representation in the original model must somehow be represented by the reduced number of spiking units, with the neuron potential acting as a trace of relevant features early in the input to be combined with those occurring later. In practice, F and C are similar in magnitude, so this transformation implies a significant reduction in the number of units used to represent the data from FT to only C (<< FT), which was deemed to be too constraining and hence lossy. To provide more representational resource, a multiplier of M = 8 was used on the number of neurons for each channel in the time-based input stream. Each of the MC neurons sample independently from the input vector and the membrane decay constant for each is selected independently of the others. Even with this change, MC << FT, in general, so the model requires fewer neurons and fewer compute steps than the original model.

The spiking, time-based, neurons incorporate dendritic delays so that a single feature may be distributed in time, the spread of which is determined by the number of delay timesteps in the neuron model. In keeping with our SpiNNaker background, we selected 16 timesteps as the natural range. In practice, a pattern that exactly for those delays would trigger the maximum output potential from that neuron, with any variations in the arrival of any of the spikes would lead to a diminished output potential, as discussed earlier.

It must be stated, however, that the new framework requires us to model the time course of spikes through the dendritic delays. This itself requires memory and compute time which is not present in the original application. Thus, there are pros and cons of this methodology.

It is worth noting that the data structures used to maintain the passage of spikes along the length of the dendrite are conceptually identical to those developed for the SpiNNaker machine (Rhodes et al., 2018). In that world, the structures were called *delay buffers* and, for each neuron, summed weights from each incoming spike into a numbered bin that represented the number of time steps before those spikes reached the soma (cell body) of the neuron. This accumulation of the contributions from many spikes saves considerable resources but relies on the fact that the contributions from individual spikes sum linearly.

3.9. Current status and ongoing work

Our experiments with test cases and the NSNet2 application are ongoing. Our training data and test data for the speech denoising test case is probably out of date.

Current observations are that performance is consistently not as good as the original models in all cases we have tried, but there are many aspects of the models that can be changed that have not yet been tried. The most obvious candidates for exploration are:

1. **Wider range of membrane potential decay time constants.** This permits correlations between input features to be detected despite being separated by a larger time difference. Increasing range implies more neurons with longer time constants, which have coarser sensitivity, but greater temporal range.

2. **Change in the distribution of time constants.** A normal distribution of time constants has been assumed, but this may be sub-optimal. A uniform distribution, for example, would increase the number of detectors at the extremes (signaling either near simultaneously occurring features or the presence of features while largely insensitive to the time delay between them) at a cost of less resolution for detectors flagging mid-range time separation. This may be a problem or even a layer-dependent choice.

3. **More time delay bins.** The number of distinct time bins into which a spike may fall creates an upper bound on the number of enumerable input features that can be detected by one neuron.

4. **Increased neurons per channel (M) in a layer.** Extra neurons imply that the expressive power of the layer is enhanced, potentially increasing the ability of that layer to extract useful features, but at a cost in extra memory for the weights and parameters and extra compute time. More subtly, beyond a critical point the extra neurons can lead to overfitting and poor generalization, an effect that has been well studied for a variety of neural networks. Ideally, we would like to find a way of determining the value of M in advance of learning, but a trial-and-error approach has been the only effective means to date.

The intention is to achieve a sufficient level of performance by the end of the CONVOLVE project to demonstrate that this approach is worth pursuing for spiking networks, and more

fundamentally that spiking networks themselves are worth pursuing as an energy-efficient alternative to traditional artificial neural networks.

4. SNN demonstrators for target applications (BOS, FMI)

4.1. Acoustic scene analysis (BOS)

In the scope of the CONVOLVE project, Bosch investigated spiking neural networks for performing acoustic scene analysis by solving two tasks. More specifically, an auditory anomaly detection use case has been considered as well as the spatial tracking of a moving sound source. For both of these tasks, SNNs have been investigated and compared with ANN models.

4.1.1. Auditory anomaly detection

For auditory anomaly detection, Bosch developed an equivalent spiking neural network for the task of siren sound detection. In more detail, a **recurrently connected network of LIF and LI neurons** has been trained by utilizing surrogate gradient methods (Neftci et al., 2019). Figure 4.1 illustrates the RNN baseline architecture, and its spiking version. The proposed spiking model has been compared to the ANN solution in terms of performance and model size (Kshirasagar, Cramer, et al., 2024), further summarized in Table 4.1.

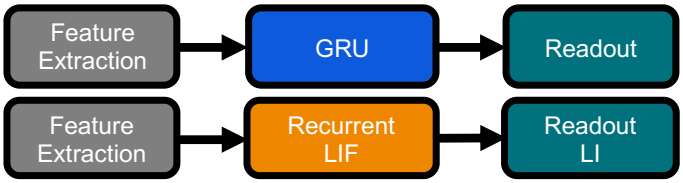


FIGURE 4.1: SCHEMATIC ILLUSTRATION OF THE GRU AND SNN ARCHITECTURE USED FOR THE DETECTION OF SIREN SOUNDS. FOR THE SNN IMPLEMENTATION, THE GRU IS REPLACED BY A RECURRENTLY CONNECTED LIF LAYER, AND A LEAKY INTEGRATOR IS USED AS A READOUT.

So far, the SNN model has been demonstrated in software simulations. We plan to map our SNN network in a collaboration with UMAN to their FPGA implementation for further benchmarking. For that purpose, the SNN needs to be quantized.

TABLE 4.1: MODEL COMPARISON AND BENCHMARKING RESULTS FOR SIREN SOUND DETECTION MODELS.

Model[1]	#Parameter	Accuracy[2]
GRU	49.8K	96.3%
SNN	16.5K	90.1%

Moreover, the parametrization of the signal processing-based feature extraction has been explored in combination with the SNN model considering performance and onset latency (Kshirasagar, Guntoro, et al., 2024). In this regard, our work further explored the sliding window approach by investigating windows with and without overlap, different mel channels and data enrichment to optimize performance. Moreover, power-of-two window sizes have been utilized to facilitate efficient hardware implementations by exploiting butterfly-like calculations.

[1] Models are implemented in PyTorch (Ansel et al., 2024) using TorchAudio (Yang et al., 2022; Hwang et al., 2023) and Norse (C.-G. Pehle & Pedersen, 2021).
[2] Trained & evaluated on artificial sequences constructed from the sound samples of the dataset presented in (Asif et al., 2022).

Our experimental results demonstrate improvements in performance accuracy by nearly 2.3% for higher overlap ratios in smaller window regimes for mel sizes of 32 and 64. Interestingly, our research shows that finetuning of neural temporal dynamics further enhances the performance of the SNN model for auditory anomaly detection. For smaller time constants and smaller window sizes, accuracy and onset latency were 5.5% higher and 2.4x lower respectively than for higher time constants. The best performing accuracies were in the range of 1–4 s for membrane and synaptic time constants.

To further validate the robustness of the spiking model for detection of siren sounds, we evaluated by enriching the siren dataset consisting of siren sounds, and various traffic noises with a bird dataset (Stowell et al., 2018) with diverse audio recordings ranging from bird to insect calls. We gathered experimental results for different mel channels on these two datasets in terms of performance accuracy. Our results confirm the ability of the spiking model to generalize to new unseen samples.

In summary, through various experiments we confirm that increasing spectral resolution generally helps the SNN model to achieve better performance (accuracy), at least up to 128 channels within this experimental scope. Further fine-tuning overlap ratios and neuronal time constants could provide a slight incremental boost to performance and onset latency.

4.1.2. Sound source tracking

For the use case of sound source tracking, Bosch considered a **spiking convolutional architecture**. More specifically, this architecture is obtained from the reduced baseline convolutional network shown at the top of Figure 4.2 by replacing ReLUs with LIF layers resulting in the spiking architecture illustrated at the bottom of Figure 4.2. In turn, the CNN and SNN versions of the sound source tracking networks could easily be compared due to their close alignment. To facilitate a mapping of the proposed SNN to the Mega accelerator in WP7, a linear leak term has been considered as a first step to match the software simulations to the dynamics implemented on the actual accelerator.

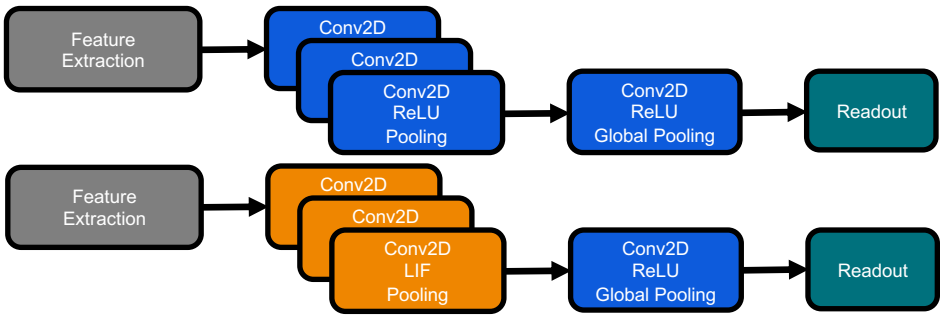


FIGURE 4.2: SCHEMATIC ILLUSTRATION OF THE CONVOLUTIONAL AND SPIKING CONVOLUTIONAL ARCHITECTURE FOR THE TRACKING OF SOUND SOURCES. RELUS ARE REPLACED BY LIF LAYERS.

Both architectures have been **benchmarked in software simulations**[3] based on speech data[4]. A comparison of the models is provided in Table 4.2 that highlights the results for the tracking of a single sound source (no noise sources are included). Again, the SNN has been trained by utilizing surrogate gradient methods (Neftci et al., 2019). At similar parameter count, the CNN model only slightly outperforms the SNN version. It is noteworthy, however, that the spike rate of this

baseline SNN still shows a slow drift to lower values at the end of the presented training run. Hence, for a detailed analysis of the spike count, even longer training runs would be required. In general, the sparse spiking of the SNN promises energy efficient implementations on dedicated accelerators.

TABLE 4.2: MODEL COMPARISON AND BENCHMARKING RESULTS FOR SOUND SOURCE TRACKING NETWORKS. THE MODELS ARE DESIGNED TO EXHIBIT A COMPARABLE PARAMETER COUNT. REPORTED MAE VALUES CORRESPOND TO THE MEDIAN, AND THE ERRORS TO THE 0.25 AND 0.75 QUANTILES CALCULATED OVER PREDICTED FRAMES. THE QUANTIZATION IS LIMITED TO THE NEURAL NETWORK RELATED CALCULATIONS, I.E. THE CALCULATIONS INVOLVED IN THE EXTRACTION OF FEATURES HAVE NOT BEEN QUANTIZED.

Model[3]	#Parameter	#Spikes	MAE [°][4]			
			FLOAT32		INT8	
CNN	92.9k	NA	$1.1^{+2.4}_{-0.8}$	$1.6^{+2.4}_{-0.8}$	$1.3^{+2.6}_{-0.6}$	$1.8^{+2.6}_{-0.9}$
SNN		4178^{+4312}_{-4052}	$1.6^{+3.1}_{-0.7}$	$2.3^{+2.4}_{-1.1}$	$2.1^{+3.7}_{-1.0}$	$2.5^{+3.7}_{-1.2}$

The aforementioned software simulation targets an implementation on the Mega accelerator. Since this accelerator supports the handling of sparse activations, the spike count of the SNN model is further reduced to foster an **energy efficient solution** after deployment. The result of this optimization is shown in Figure 4.3. Here, the median number of spikes required to predict a single frame is reduced with only a minimal loss in performance. For $\lambda_{reg} > 5.0e-10$, the training tends to become unstable.

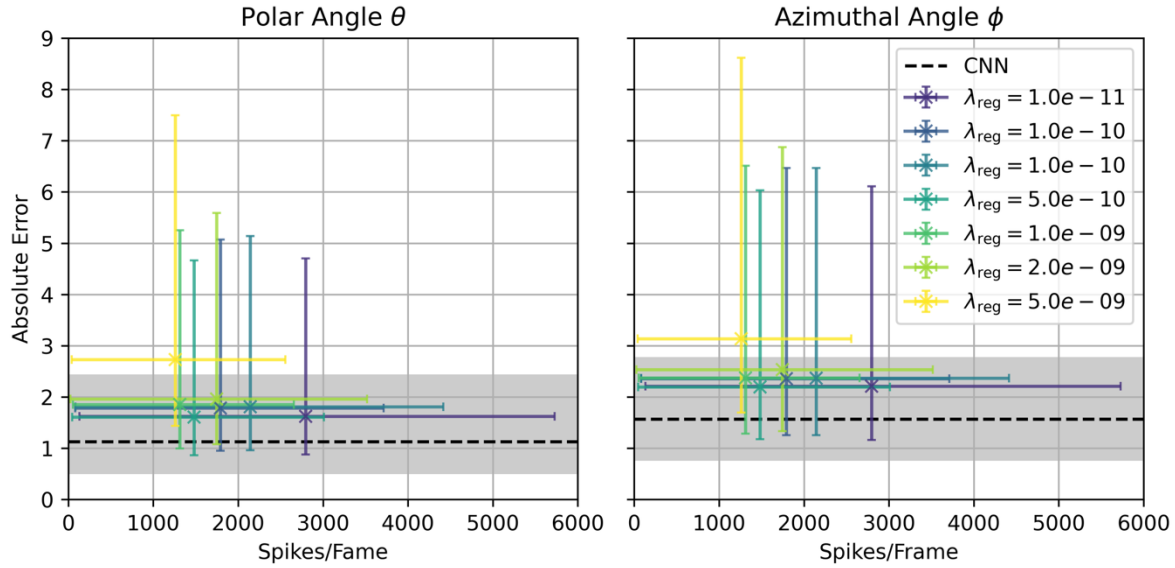


FIGURE 4.3: THE NUMBER OF SPIKES REQUIRED TO PREDICT A SINGLE FRAME COULD BE REDUCED FOR THE SOUND SOURCE TRACKING NETWORK. ILLUSTRATION OF THE RESULTS FOR BOTH PREDICTED ANGLES AND DIFFERENT SPIKE COUNT REGULARIZATION STRENGTHS AS WELL AS THE CNN RESULT FOR COMPARISON. REPORTED VALUES CORRESPOND TO THE MEDIAN AND THE ERRORS TO THE 0.25 AND 0.75 QUANTILES CALCULATED OVER PREDICTED FRAMES. FOR THE CNN BASELINE, THE MEDIAN OF THE ERROR IS INDICATED BY A BLACK DASHED LINE AND THE QUANTILES BY A GREY AREA.

The **feasibility of mapping** the sound source tracking network to the Mega accelerator is currently under investigation in a collaboration with TU/e. In this process, further

[3] Models are implemented in PyTorch Lightning (Falcon & The PyTorch Lightning team, 2024) and PyTorch (Ansel et al., 2024) using TorchAudio (Yang et al., 2022; Hwang et al., 2023).
[4] Trained & evaluated on a subset of the sound samples of Librispeech (Panayotov et al., 2015) that have been augmented with the tool pyroadacoustics (Damiano, 2022) to simulate propagation effects. For each sample, only a single sound source has been simulated, i.e. no sources of noise have been included.

characteristics of the hardware implementation as well as the employed quantization will be included into the software simulations to obtain a more detailed performance estimate.

4.2. Speech quality prediction (FMI)

4.2.1. Outline of application

The FMI has developed an SNN model for speech quality prediction (SQP) (Nilsson et al., 2025) in collaboration with GNA for their use-case of speech enhancement with dynamic neural networks (DyNNs). SQP is to be used for decision-making in the dynamic routing of the computational graph in such models, such as in early exiting or dynamic channel pruning (Miccini et al., 2023, 2025). The present work presents the first published SNN for SQP and compares it against existing lightweight ANN baselines based on CNN or LSTM (Fu et al., 2018; Reddy et al., 2021). The developed model, benefitting from the natively streaming processing with sparse activations of SNNs, achieves comparable performance to the ANN baselines while reducing the computational cost by at least 2.5x, assuming hardware that supports sparse activations.

SQP is a regression task that typically consists of predicting one speech quality score per labeled audio sample. Here, the Interspeech DNS Challenge 2020 dataset (Reddy et al., 2020) was used, from which 5-second samples of audiobook readings with synthetically added noise were labeled with the PESQ speech quality metric. While conventional metrics like PESQ are "intrusive"—i.e., they need a clean reference signal for their computation—deep learning methods enable effective non-intrusive SQP directly from noisy audio.

4.2.2. Model

The developed SNN model (Figure 4.4) follows a standard audio processing pipeline in taking log-Mel power spectrograms as inputs. As the SNN is a causal, stateful model, it takes one spectrogram frame—one sequence step—at a time and is thus naturally suited for streaming inference. The spectrogram data is converted to spikes by being fed as analog currents into a layer of adaptive leaky integrate-and-fire (adLIF) neurons, which are included in the end-to-end training. This encoder layer is followed by a number of layers of leaky integrate-and-fire (LIF) neurons, which use 1D convolutions along the frequency dimensions of the data, while temporal information is integrated in the neuronal states. The convolutional motif grants the same kind of parameter efficiency as in conventional CNNs. Finally, a non-spiking output unit provides a continuous readout of the network's quality estimate.

Two versions of the model were investigated: a feedforward version (ff-CSNN), and a version in which the convolutional layers are also recurrently connected (r-CSNN). The models were trained with backpropagation through time (BPTT) using standard surrogate gradient descent (Neftci et al., 2019). During training and evaluation, the mean over time of the readout signal was taken as the estimate of the single PESQ quality score of a sample.

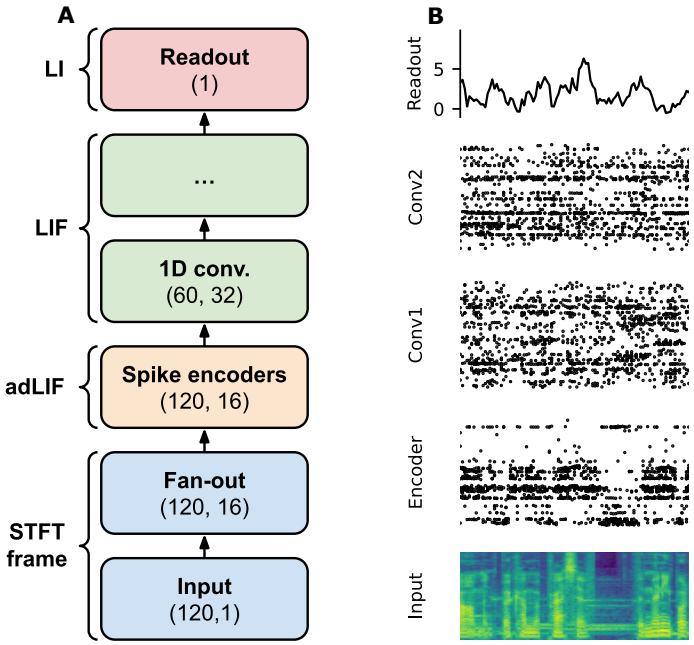


FIGURE 4.4: (A) CONVOLUTIONAL SNN ARCHITECTURE FOR SPEECH QUALITY PREDICTION. THE DIMENSIONS CORRESPOND TO THE PROCESSING OF ONE FRAME IN ONE SNN TIME-STEP. THE NUMBERS IN PARENTHESES INDICATE THE SPECTRAL AND CHANNEL OUTPUT DIMENSIONS, RESPECTIVELY. THE NUMBER OF LIF NEURONS IN EACH CONVOLUTIONAL LAYER CORRESPONDS TO ITS NUMBER OF OUTPUT ACTIVATIONS. (B) EXAMPLE NETWORK ACTIVITY DURING 2 S.

4.2.3. Handling temporal sparsity

Temporal activation sparsity (i.e. spike sparsity) was investigated by including spike-rate regularization in the BPTT training. An L2 upper-bound regularization term on population-level average spike-counts was employed to achieve different degrees of activation sparsity. Without such regularization, both the recurrent and feedforward versions of the model have a default activation sparsity of about 94%—i.e., on average, 94% of the neurons are not spiking in a given time step. This quite high default sparsity is likely due to the convolutional architecture, in which there is a relatively high number of neurons, each having few input connections.

4.2.4. Demonstrator

The FMI has developed a demonstrator for the SQP SNN, in which inference is run in real time on a laptop. The demonstrator works in a loop that takes audio from the laptop microphone, computes a spectrogram frame, and finally computes a forward pass with the SNN, which retains its state between loop iterations. In order to make the inference pass fast enough for seamless real-time operation, the SNN—originally trained using PyTorch—was implemented in JAX, which brought down the inference time to a few milliseconds alongside the 20 ms loop iteration latency inherited from the spectrogram frame length.

Furthermore, MAN is currently working on a hardware implementation of the SQP SNN on an FPGA-based accelerator.

4.2.5. Results

The Pearson correlation coefficient (PCC) is used as the primary SQP performance metric. The results presented in Table 4.3 show how the r-CSNN performs better than the ff-CSNN and almost on par with the streaming baseline model Quality-Net based on unidirectional LSTM. These models are outperformed by DNSMOS, which is based on non-causal 2D convolutions. This CNN architecture would, however, give rise to substantial overhead from overlapping multi-second input windows in a streaming setting.

Figure 4.5 shows how the SNN's task performance in terms of the PCC is affected by increasing activation sparsity. These results show that the sparsity can be increased from 94% to about 99% with moderate impact on performance, after which performance drops sharply. The two versions of the SNN show similar behavior in this regard, with the recurrent version having both somewhat higher performance overall as well as a slightly higher robustness to high degrees of sparsity.

The computational cost of the models was estimated in terms of effective floating-point operations (EFLOPs) (Narduzzi et al., 2025), see Table 4.3, for which operations involving zeros are ignored based on the assumption of hardware that supports activation sparsity. EFLOPs were measured using NeuroBench (Yik et al., 2025). The "static" EFLOPs numbers stemming from the constant simulation cost per neuron and time-step for the SNNs presented here are updated with a slight correction from those published in the paper. Here, correctly ignoring spike-sparsity dependent terms as well as assuming folding of static scaling terms into the weights, we counted, per time-step, three FLOPs per LIF neuron and seven per adLIF neuron instead of five and 11, which were used in the paper. This update results in somewhat lower total EFLOPs for the SNNs.

Both CSNN variants achieve significant efficiency gains over the baseline models. Without enforced activation sparsity, the r-CSNN uses about 2.5x less EFLOPs than the LSTM-based model, and the ff-CSNN uses 3x less. The LSTM-based model in turn uses about half the EFLOPs of the CNN per sample—an estimate that does not yet include the CNN's computational overhead due to redundancy of its sliding window in the streaming data setting, which would be added on top. Using the operating point at 99.3% activation sparsity of the r-CSNN brought down its EFLOPs to 5x less than the LSTM-based model at the price of a moderate performance degradation, which may be acceptable for resource-constrained applications. While the spike-driven, dynamic EFLOPs are larger than the neuron-state driven, static EFLOPs by default, in the sparse operating regime, the static EFLOPs are almost 3x larger.

TABLE 4.3: TASK PERFORMANCE AND COMPUTATIONAL COST ON TEST SET. PERFORMANCE METRICS ARE REPORTED AS MEAN ± STANDARD DEVIATION ACROSS FIVE RUNS. THE NUMBERS OF EFLOPS ARE PER 5-S INPUT SAMPLE, WITH DYNAMIC EFLOPS AVERAGED OVER THE TEST SET. BOLD FONT INDICATES BEST RESULTS.

	Model	Streaming	Spiking	PCC ↑	MSE ↓	EFLOPs			Params.
						Static	Dynamic	Total	
Baselines	DNSMOS (CNN)	✗	✗	**0.865 ± 0.008**	0.272 ± 0.024	–	93.2 M	93.2 M	46 k
	Quality-Net[1] (LSTM)	✓	✗	0.830 ± 0.014	0.203 ± 0.025	44.2 M	–	44.2 M	96 k
Our models	r-CSNN	✓	✓	0.810 ± 0.002	**0.179 ± 0.005**	6.22 M	11.3 M	17.5 M	56 k
	r-CSNN (99.3 % sparsity)	✓	✓	0.782 ± 0.009	0.206 ± 0.006	6.22 M	2.25 M	**8.47 M**	56 k
	ff-CSNN	✓	✓	0.798 ± 0.010	0.206 ± 0.013	6.24 M	8.57 M	14.8 M	**39 k**

[1]Causal reimplementation based on unidirectional LSTM.

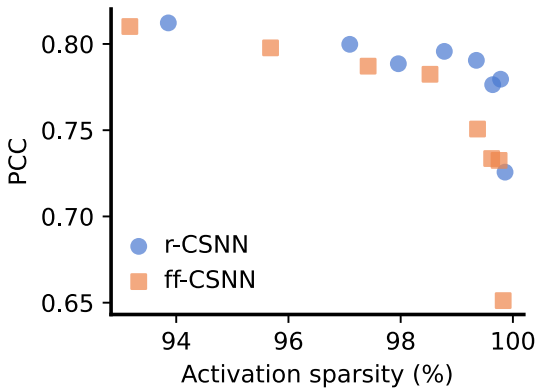


FIGURE 4.5: PERFORMANCE OF CSNN MODELS ON TEST SET AT DIFFERENT LEVELS OF ACTIVATION SPARSITY.

4.2.6. Challenges and limitations

While the CSNNs natively support streaming inference, they were trained in a conventional SQP setup. This means that labels are provided at the level of samples, i.e., with a coarse temporal resolution, as SQP is typically based on human mean-opinion-scores (MOS) or objective metrics like PESQ. Here, we simply let the mean over time per sample of the continuous network readout represent its PESQ estimate. Future work targeting streaming SQP could explore inclusion of time-varying objective metrics, such as frame-wise signal-to-distortion ratio (SDR), to evaluate and improve the precision of predictions on finer timescales.

The efficiency analysis revealed that neuronal state updates contribute more than spike-driven operations to the computational cost in sparsely spiking CSNNs. This suggests that future work should explore reducing the density of neuronal state computations. This could for instance be achieved by using approximation schemes for neuronal state updates (Wang et al., 2016; Eissa et al., 2021). However, this only applies to digital hardware implementations of SNNs, as on mixed-signal neuromorphic hardware, analog electronics are used to emulate neuronal dynamics at low cost (C. Pehle et al., 2022; Richter et al., 2024). Such hardware implementations of the models could be substantially more efficient than what the digital-hardware estimates presented here imply.

5. Online learning strategies (FMI)

5.1. Online learning algorithm

5.1.1. Overview

The FMI is conducting ongoing work on a temporally local online learning algorithm for SNNs. This work builds on existing literature in which the implicit recurrence of the stateful neurons of SNNs is leveraged to make real-time recurrent learning (RTRL) computationally tractable (Zenke & Neftci, 2021; Lohoff et al., 2025). In feedforward SNNs, mixed-mode RTRL becomes mathematically equivalent to learning with so-called synaptic eligibility traces in combination with temporally local, spatial backpropagation (Zenke & Ganguli, 2018; Bellec et al., 2020; Eshraghian et al., 2023).

Due to the "implicit" self-recurrence of feedforward SNNs being diagonal, the compute scaling of RTRL with respect to the average number of neurons per layer, n, is reduced to $O(n^2)$ from the $O(n^4)$ of densely recurrent networks. Still, the synaptic eligibility traces—being first-order low-pass filters of temporally correlated pre- and post-synaptic events—add one decaying, continuous state per synapse in addition to the existing decaying neuron states. While such continuous states can be implemented at low cost in sub-threshold analog circuits, their implementation in digital hardware requires the explicit re-evaluation of every state in each simulated time step, which is computationally costly.

The FMI's ongoing work investigates the use of simplifying approximations to the continuous states involved in RTRL with SNNs to reduce the number of computations needed for state updates in each time step. Specifically, the removal of the decay mechanisms of the neuron membranes and synaptic eligibility traces or their replacement with timed state-resets is investigated. Other simplifications of the involved computations are investigated, such as binarizing the surrogate spike-derivative, which is used in computing the eligibility traces.

5.1.2. Preliminary results and future work

So far, the FMI have investigated their learning algorithm for the basic case of a feedforward SNN with a single hidden layer (1024 LIF neurons) on the standard SNN audio benchmark SHD (Cramer et al., 2022). Table 5.1 presents the resulting accuracy of this SNN when trained with BPTT, RTRL, or the developed algorithm in which all neuronal and eligibility-trace decay mechanisms are removed and the surrogate spike-derivative is binarized. These results indicate that the investigated approximations to RTRL cause only a moderate decrease in performance. While BPTT and RTRL should in principle compute the same gradient, there is a difference in the presented results in favor of RTRL, which may stem from slight implementation differences.

TABLE 5.1: TEST ACCURACY ON SHD FOR DIFFERENT SNN LEARNING ALGORITHMS. THE ACCURACY IS REPORTED AS MEAN ± STANDARD DEVIATION ACROSS FIVE RUNS.

Algorithm	Accuracy
BPTT	0.728 ± 0.009
RTRL	0.756 ± 0.015
Ours	0.733 ± 0.009

Future work will include investigating these approximations on more powerful architectures with explicit recurrence and multiple hidden layers. For this, explicit (non-diagonal) recurrent connections need to be detached from the gradient computations to maintain the tractable $O(n^2)$ compute scaling of RTRL discussed above. Furthermore, so far, the loss function and gradient updates were computed on a per-sequence basis, as is standard for SNN training with BPTT. Investigating the impact on performance and possible efficiency benefits of moving to a per-step loss and gradient computation is subject to future work. Finally, the efficiency benefits of the learning algorithm will be quantitatively estimated, e.g. in terms of sparsity-aware effective floating-point operations (EFLOPs) (Narduzzi et al., 2025) and memory requirements.

5.2. Handling temporal sparsity

In the FMI's development of an SNN for speech quality prediction, temporal activation-sparsity (i.e. sparse spikes) was investigated through spike-count regularization in BPTT training; see Sec. 4.2 for details and results.

6. Conclusions and future work

6.1. Summary of key achievements

The key achievements presented in this deliverable are SNN models developed within CONVOLVE for three different auditory target applications: auditory anomaly detection, sound source tracking, and speech quality prediction (SQP). These SNNs were trained end-to-end with surrogate gradient descent (Neftci et al., 2019), which fits SNNs into the framework of deep learning and has become the standard method for developing functional SNNs (Eshraghian et al., 2023).

The SNN for anomaly detection performs similarly to a GRU-based model with similar topology, while using 3x fewer parameters. An analysis of the impact of input-window length and neuronal time constants on this model's accuracy and onset latency was conducted. Hardware implementations of the model on Raspberry Pi 4 as well as on MAN's FPGA are under development.

The SNN for sound source tracking performs similarly to a baseline CNN with an equal number of parameters and is being investigated for implementation on the SNN accelerator Mega by TUE.

The SNN for SQP performs comparatively to existing lightweight models based on CNN or LSTM while reducing the estimated number of FLOPs by 2.5–5x depending on the degree of spike-sparsity, which comes at a characterized trade-off with task performance. A demonstrator for the model was developed, in which real-time streaming inference is run on a laptop. Furthermore, a hardware implementation on an FPGA-based accelerator is under development by MAN.

In conclusion, all three developed SNNs perform close to but slightly below conventional ANNs while bringing potential efficiency benefits. While these efficiency benefits remain to be assessed in hardware implementations, the efficiency estimate made for the SQP SNN suggests that the original goal of Task T4.3 of a >3x efficiency increase over ANNs was reached. In terms of the goal of a generalized methodology, the developed SNNs converged on using surrogate-gradient training and LIF neurons in combination with conventional circuit motifs from deep learning like convolutions and recurrence.

6.2. Remaining challenges and open questions

Remaining work associated with this deliverable includes finalizing hardware implementations of the developed SNN models and quantifying final efficiency gains. Implementing the models in

hardware may involve challenges related to quantization and fixed-point state representations and assessing possible related performance decreases.

As revealed in the FMI's efficiency analysis for SQP, neuronal state updates become the largest compute component in spike-sparse regimes of CSNNs. Future work should investigate approximations of state dynamics (Wang et al., 2016; Eissa et al., 2021) to reduce the associated cost on digital hardware. Such approximations are being investigated in the FMI's ongoing development of an online learning algorithm. Furthermore, the SQP model development was based on a standard task setup in which there is a single quality label per multi-second audio sample. Therefore, a remaining challenge for streaming SQP is to assess and ensure the fidelity of the quality predictions on finer timescales.

For the work on online learning with approximate SNN dynamics, remaining challenges include the application of the approximation scheme on recurrent and deep architectures and a quantitative estimate of the associated compute and memory savings.

7. References

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., … Chintala, S. (2024). PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 929–947. https://doi.org/10.1145/3620665.3640366

Asif, M., Usaid, M., Rashid, M., Rajab, T., Hussain, S., & Wasi, S. (2022). Large-scale audio dataset for emergency vehicle sirens and road noises. *Scientific Data*, *9*(1), 599. https://doi.org/10.1038/s41597-022-01727-2

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, *11*(1), Article 1. https://doi.org/10.1038/s41467-020-17236-y

Bos, H., & Muir, D. (2023). Sub-mW Neuromorphic SNN Audio Processing Applications with Rockpool and Xylo. In *Embedded Artificial Intelligence*. River Publishers.

Bos, H., & Muir, D. R. (2024). *Micro-power spoken keyword spotting on Xylo Audio 2* (No. arXiv:2406.15112). arXiv. http://arxiv.org/abs/2406.15112

Branco, T., & Häusser, M. (2010). The single dendritic branch as a fundamental functional unit in the nervous system. *Current Opinion in Neurobiology*, *20*(4), 494–502. https://doi.org/10.1016/j.conb.2010.07.009

Cheng, H., Zhang, M., & Shi, J. Q. (2024). A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), 10558–10578. https://doi.org/10.1109/TPAMI.2024.3447085

Cramer, B., Stradmann, Y., Schemmel, J., & Zenke, F. (2022). The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(7), 2744–2757. https://doi.org/10.1109/TNNLS.2020.3044364

Cumlin, F., Liang, X., Ungureanu, V., K. A. Reddy, C., Schüldt, C., & Chatterjee, S. (2024). DNSMOS Pro: A Reduced-Size DNN for Probabilistic MOS of Speech. *Interspeech 2024*, 4818–4822. https://doi.org/10.21437/Interspeech.2024-478

Damiano, S. (2022). Pyroadacoustics: A Road Acoustics Simulator Based on Variable Length Delay Lines. *Proceedings of the 25th International Conference on Digital Audio Effects (DAFx20in22)*.

Davidson, S., & Furber, S. B. (2021). Comparison of Artificial and Spiking Neural Networks on Digital Hardware. *Frontiers in Neuroscience*, *15*, 651141. https://doi.org/10.3389/fnins.2021.651141

Eissa, S., Stuijk, S., & Corporaal, H. (2021). Hardware Approximation of Exponential Decay for Spiking Neural Networks. *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 1–4. https://doi.org/10.1109/AICAS51828.2021.9458560

Eshraghian, J. K., Ward, M., Neftci, E. O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D. S., & Lu, W. D. (2023). Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE*, *111*(9), 1016–1054. Proceedings of the IEEE. https://doi.org/10.1109/JPROC.2023.3308088

Falcon, W., & The PyTorch Lightning team. (2024). *PyTorch Lightning* (Version 2.4.0) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.13254264

Farina, M., Ahmad, U., Taha, A., Younes, H., Mesbah, Y., Yu, X., & Pedrycz, W. (2024). Sparsity in transformers: A systematic literature review. *Neurocomputing*, *582*, 127468. https://doi.org/10.1016/j.neucom.2024.127468

Fu, S., Tsao, Y., Hwang, H.-T., & Wang, H.-M. (2018). Quality-Net: An End-to-End Non-intrusive Speech Quality Assessment Model Based on BLSTM. *Interspeech 2018*, 1873–1877. https://doi.org/10.21437/Interspeech.2018-1802

Hao, X., Ma, C., Yang, Q., Wu, J., & Tan, K. C. (2025). Toward Ultralow-Power Neuromorphic Speech Enhancement With Spiking-FullSubNet. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. https://doi.org/10.1109/TNNLS.2025.3566021

Hwang, J., Hira, M., Chen, C., Zhang, X., Ni, Z., Sun, G., Ma, P., Huang, R., Pratap, V., Zhang, Y., Kumar, A., Yu, C.-Y., Zhu, C., Liu, C., Kahn, J., Ravanelli, M., Sun, P., Watanabe, S., Shi, Y., & Tao, Y. (2023). TorchAudio 2.1: Advancing Speech Recognition, Self-Supervised Learning, and Audio Processing Components for Pytorch. *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1–9. https://doi.org/10.1109/ASRU57964.2023.10389648

Izhikevich, E. M. (2006). Polychronization: Computation with Spikes. *Neural Computation*, *18*(2), 245–282. https://doi.org/10.1162/089976606775093882

Kshirasagar, S., Cramer, B., Guntoro, A., & Mayr, C. (2024). Auditory Anomaly Detection using Recurrent Spiking Neural Networks. *2024 IEEE 6th International Conference on AI*

Circuits and Systems (AICAS), 278–281. https://doi.org/10.1109/AICAS59952.2024.10595878

Kshirasagar, S., Guntoro, A., & Mayr, C. (2024). Impact of Sliding Window Variation and Neuronal Time Constants on Acoustic Anomaly Detection Using Recurrent Spiking Neural Networks in Automotive Environment. *Algorithms*, *17*(10), Article 10. https://doi.org/10.3390/a17100440

Kudithipudi, D., Schuman, C., Vineyard, C. M., Pandit, T., Merkel, C., Kubendran, R., Aimone, J. B., Orchard, G., Mayr, C., Benosman, R., Hays, J., Young, C., Bartolozzi, C., Majumdar, A., Cardwell, S. G., Payvand, M., Buckley, S., Kulkarni, S., Gonzalez, H. A., … Furber, S. (2025). Neuromorphic computing at scale. *Nature*, *637*(8047), 801–812. https://doi.org/10.1038/s41586-024-08253-8

Li, C. (2022). *Optimizing Rate-Based Spiking Neural Networks* [University of Manchester]. https://research.manchester.ac.uk/en/studentTheses/optimizing-rate-based-spiking-neural-networks

Li, Z., & Li, W. (2023). MOSLight: A Lightweight Data-Efficient System for Non-Intrusive Speech Quality Assessment. *INTERSPEECH 2023*, 5386–5390. https://doi.org/10.21437/Interspeech.2023-263

Lohoff, J., Kaya, A., Assmuth, F., & Neftci, E. (2025). A Truly Sparse and General Implementation of Gradient-Based Synaptic Plasticity. *2025 Neuro Inspired Computational Elements (NICE)*, 1–9. https://doi.org/10.1109/NICE65350.2025.11065510

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7

Miccini, R., Laroche, C., Piechowiak, T., & Pezzarossa, L. (2025). Scalable Speech Enhancement With Dynamic Channel Pruning. *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49660.2025.10889876

Miccini, R., Zniber, A., Laroche, C., Piechowiak, T., Schoeberl, M., Pezzarossa, L., Karrakchou, O., Sparsø, J., & Ghogho, M. (2023). Dynamic nsNET2: Efficient Deep Noise Suppression with Early Exiting. *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. https://doi.org/10.1109/MLSP55844.2023.10285925

Narduzzi, S., Zenke, F., Liu, S.-C., & Dunbar, L. A. (2025). EFLOP: A sparsity-aware metric for evaluating computational cost in spiking and non-spiking neural networks. *Neuromorphic Computing and Engineering*, *5*(3), 034011. https://doi.org/10.1088/2634-4386/addee8

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine*, *36*(6), 51–63. https://doi.org/10.1109/MSP.2019.2931595

Nilsson, M., Miccini, R., Rossbroich, J., Laroche, C., Piechowiak, T., & Zenke, F. (2025). *Efficient Streaming Speech Quality Prediction with Spiking Neural Networks*. 5423–5427. https://doi.org/10.21437/Interspeech.2025-2269

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964

Pehle, C., Billaudelle, S., Cramer, B., Kaiser, J., Schreiber, K., Stradmann, Y., Weis, J., Leibfried, A., Müller, E., & Schemmel, J. (2022). The BrainScaleS-2 Accelerated Neuromorphic System With Hybrid Plasticity. *Frontiers in Neuroscience*, *16*. https://doi.org/10.3389/fnins.2022.795876

Pehle, C.-G., & Pedersen, J. E. (2021). *Norse—A deep learning library for spiking neural networks* (Version 0.0.5)[Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.4422025

Reddy, C. K. A., Gopal, V., & Cutler, R. (2021). Dnsmos: A Non-Intrusive Perceptual Objective Speech Quality Metric to Evaluate Noise Suppressors. *ICASSP 2021 - 2021 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6493–6497. https://doi.org/10.1109/ICASSP39728.2021.9414878

Reddy, C. K. A., Gopal, V., Cutler, R., Beyrami, E., Cheng, R., Dubey, H., Matusevych, S., Aichner, R., Aazami, A., Braun, S., Rana, P., Srinivasan, S., & Gehrke, J. (2020). The INTERSPEECH 2020 Deep Noise Suppression Challenge: Datasets, Subjective Testing Framework, and Challenge Results. *Interspeech 2020*, 2492–2496. https://doi.org/10.21437/Interspeech.2020-3038

Rhodes, O., Bogdan, P. A., Brenninkmeijer, C., Davidson, S., Fellows, D., Gait, A., Lester, D. R., Mikaitis, M., Plana, L. A., Rowley, A. G. D., Stokes, A. B., & Furber, S. B. (2018). sPyNNaker: A Software Package for Running PyNN Simulations on SpiNNaker. *Frontiers in Neuroscience*, *12*, 816. https://doi.org/10.3389/fnins.2018.00816

Richter, O., Wu, C., Whatley, A. M., Köstinger, G., Nielsen, C., Qiao, N., & Indiveri, G. (2024). DYNAP-SE2: A scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor. *Neuromorphic Computing and Engineering*, *4*(1), 014003. https://doi.org/10.1088/2634-4386/ad1cd7

Rueckauer, B., & Liu, S.-C. (2018). Conversion of analog to spiking neural networks using sparse temporal coding. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. https://doi.org/10.1109/ISCAS.2018.8351295

Stowell, D., Pamuła, H., & Wood, M. (2018). *Evaluation Datasets For Dcase 2018 Bird Audio Detection* [Dataset]. Zenodo. https://doi.org/10.5281/ZENODO.1298603

Sun, T., & Bohté, S. (2024). DPSNN: Spiking neural network for low-latency streaming speech enhancement. *Neuromorphic Computing and Engineering*, *4*(4), 044008. https://doi.org/10.1088/2634-4386/ad93f9

Wang, R., Thakur, C. S., Hamilton, T. J., Tapson, J., & van Schaik, A. (2016). A stochastic approach to STDP. *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2082–2085. https://doi.org/10.1109/ISCAS.2016.7538989

Yan, Y., Yang, Q., Wu, Y., Liu, H., Zhang, M., Li, H., Tan, K. C., & Wu, J. (2025). Efficient and robust temporal processing with neural oscillations modulated spiking neural networks. *Nature Communications*, *16*(1), 8651. https://doi.org/10.1038/s41467-025-63771-x

Yang, Y.-Y., Hira, M., Ni, Z., Astafurov, A., Chen, C., Puhrsch, C., Pollack, D., Genzel, D., Greenberg, D., Yang, E. Z., Lian, J., Hwang, J., Chen, J., Goldsborough, P., Narenthiran, S., Watanabe, S., Chintala, S., & Quenneville-Belair, V. (2022). Torchaudio: Building Blocks for Audio and Speech Processing. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6982–6986. https://doi.org/10.1109/ICASSP43922.2022.9747236

Yik, J., Van den Berghe, K., den Blanken, D., Bouhadjar, Y., Fabre, M., Hueber, P., Ke, W., Khoei, M. A., Kleyko, D., Pacik-Nelson, N., Pierro, A., Stratmann, P., Sun, P.-S. V., Tang, G., Wang, S., Zhou, B., Ahmed, S. H., Vathakkattil Joseph, G., Leto, B., … Reddi, V. J. (2025). The neurobench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, *16*(1), 1545. https://doi.org/10.1038/s41467-025-56739-4

Zenke, F., & Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, *30*(6), 1514–1541. https://doi.org/10.1162/neco_a_01086

Zenke, F., & Neftci, E. O. (2021). Brain-Inspired Learning on Neuromorphic Substrates. *Proceedings of the IEEE*, *109*(5), 935–950. Proceedings of the IEEE. https://doi.org/10.1109/JPROC.2020.3045625

8. Annexes

8.1. Links to code

8.1.1. Speech quality prediction with SNNs

https://github.com/fmi-basel/spiking-sqp