Faculty of Physics and Astronomy

University of Heidelberg

Master's Thesis

in Physics,

submitted by

Johannes Weis

born in Heppenheim, Germany

September 2020

Inference with Artificial Neural Networks on Neuromorphic Hardware

This thesis has been carried out by

Johannes Weis

at the

Kirchhoff-Institute for Physics,

Heidelberg University

under the supervision of

Dr. Johannes Schemmel

Abstract

Recent advances with artificial neural networks were partly achieved using ever larger models. To cope with the growing computational demands, new hardware architectures are explored. We use BrainScaleS-2, a mixed-signal system targeted at emulating spiking neural networks, in an inference mode for artificial neural networks. The most common operation, multiplication of a matrix with a vector, is handled in the analog core, promising good energy efficiency.

The circuitry is subject to fixed pattern deviations due to mismatch in the substrate. We present calibration algorithms which ensure well-matched operating ranges of the involved components. Characterizing the multiplication operation shows mostly linear behavior in both arguments, with distortions due to remaining fixed pattern noise and trial to trial variations in the order of 3 % each.

Benchmarking the performance on a convolutional neural network classifying the MNIST dataset of handwritten digits yields a classification accuracy of 98.0 % after training on chip, close to the original accuracy on CPU. The energy efficiency during multiply-accumulate operations is approximately 150 GOPS/W, which is less than what state-of-the-art edge devices achieve. Still, especially the combination with spiking layers on the same substrate is compelling. We explore such hybrid approaches for detecting atrial fibrillation in electrocardiogram traces.

Zusammenfassung

Fortschritte auf dem Gebiet künstlicher neuronaler Netze werden zum Teil durch immer größere Modelle erzielt. Um diese zahlreichen Rechenoperationen handhaben zu können, werden neue Hardwarekonzepte erforscht. Wir verwenden BrainScaleS-2, ein Mischsignal-System, das für die Emulation pulsbasierter neuronaler Netze konzipiert wurde, in einem Inferenzmodus für künstliche neuronale Netze. Die häufigste Operation, Multiplikation einer Matrix mit einem Vektor, wird im analogen Kern ausgeführt, was hohe Energieeffizienz verspricht.

Die Schaltungen weisen Abweichungen aufgrund von Fertigungstoleranzen auf. Wir stellen Kalibrationsroutinen vor, welche die Arbeitsbereiche der einzelnen Komponenten aufeinander abstimmen. Für die Multiplikations-Operation wurde eine meist lineare Abhängigkeit von beiden Operanden bestätigt. Die verbleibenden systematischen Abweichungen und die zufälligen Variationen betragen jeweils etwa 3 %.

Die Funktionsfähigkeit wird bei der Klassifikation des MNIST-Datensatzes handgeschriebener Ziffern bewiesen. Die dabei erzielte Genauigkeit von 98.0 % nach Training auf dem Chip entspricht nahezu der ursprünglichen Genauigkeit auf einer CPU. Die Energieeffizienz bei Multiplikations-Akkumulations-Operationen beträgt etwa 150 GOPS/W, was von anderen aktuellen Systemen bereits übertroffen wird. Dennoch birgt insbesondere die Kombination mit pulsbasierten Netzwerken Potential. Wir untersuchen solch hybride Ansätze, um Vorhofflimmern in Elektrokardiogramm-Signalen zu erkennen.

Contents

1 Introduction

Artificial neural networks are used in a wide variety of applications [LeCun, Bengio, and Hinton, 2015]. Following the principle of the perceptron [Rosenblatt, 1958], today's networks can not only recognize text and speech but also classify images, recognize persons, translate text and even generate new images or news articles [Brown et al., 2020; Gatys, Ecker, and Bethge, 2015]. The recent progress with artificial neural networks was largely due to more complex models involving more hidden layers [Schmidhuber, 2015; Bianchini and Scarselli, 2014].

Typical networks primarily consist of dense and convolution layers with non-linear activation functions. A dense layer maps a vector of input activations to a vector of output neuron activations by multiplying the weight matrix with the inputs. For convolutions, the inputs are divided into small subsets, each is multiplied with a constant weight matrix. The input tensor is therefore convolved with the weight matrix. Hence, for computation, dense and convolution layers are mainly one or multiple vector-matrix multiplication operations. Matrix multiplication is also the basis for more involved structures, for example long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] or residual networks [He et al., 2016]. Adding biases or non-linearities to the network contributes less to the number of computations, as these only operate on the results and do not involve a larger matrix.

As artificial neural networks have become larger, their computational complexity has increased. Besides hardware cost, energy consumption has become an issue [Strubell, Ganesh, and McCallum, 2019; Schwartz et al., 2019]. The cost of training some models has reached the order of millions of dollars, thus computing power imposes limits on further improvements in network performance [Thompson et al., 2020; So, Liang, and Le, 2019; Wu et al., 2020]. Since vector-matrix multiplications are the most common task during inference, an efficient implementation of the underlying multiply-accumulate (MAC) operation is desired. Due to their parallelism, graphics processing units (GPUs) already provide an advantage over general-purpose CPUs [Oh and Jung, 2004]. With built-in support of GPUs in popular machine-learning software frameworks, training on GPUs has become popular [Abadi et al., 2015; Paszke et al., 2019].

There are different approaches to the development of specialized hardware targeting efficient inference. In the space of digital processors [Yin et al., 2017], most notably, Google's Tensor Processing Unit (TPU) has shown significant improvements in computation time and energy efficiency in productive usage [Jouppi et al., 2017]. Mixed-signal circuit designs promise an even higher energy efficiency. Inputs and outputs are still digital, but the computation happens in an analog core [Boser et al., 1991]. During a MAC operation, the accumulation of all the analog results can happen naturally by collecting charge on a capacitor. In a digital design, the accumulation would require further operations. Multiple solutions implementing MAC operations in an analog fashion have been

proposed [Verleysen et al., 1994; Schemmel et al., 2004; Stefano Ambrogio et al., 2018; Kim et al., 2019; Yamaguchi et al., 2019; Cosemans et al., 2019]. These systems can handle only a small weight precision compared to CPUs or GPUs, which can be as low as binary weights. Reducing weight precision is already a commonly used approach to decrease resource usage [Micikevicius et al., 2017]. However, the network architecture has to be able to cope with the limited weight precision, which may result in larger networks [Draghici, 2002].

The BrainScaleS-2 hardware used in this thesis is mainly targeted at accelerated emulation of spiking neural networks [Schemmel et al., 2020; Billaudelle et al., 2019]. However, its analog core, consisting of a synapse matrix capable of processing plasticity and neurons following the adaptive exponential integrate-and-fire model [Brette and Gerstner, 2005], can also be used for vector-matrix multiplication. Utilizing the short term plasticity circuitry, the pulse width reaching synapses is modulated, encoding 5 bit vector entries. The 6 bit synaptic weights modulate the emitted current. The charge received on neurons is thus the product of vector entry and matrix weight. With neuron dynamics based on a leaky integrator, results are accumulated on the neurons' membrane capacitors and can be digitized with 8 bit precision. Multiply-accumulate operations can therefore be executed on the analog core of the chip [Weis et al., 2020]. Spiking operation and vector-matrix multiplication can also be mixed, allowing, e. g., an image filtered by a convolutional layer getting classified with multiple spiking layers, all within one chip.

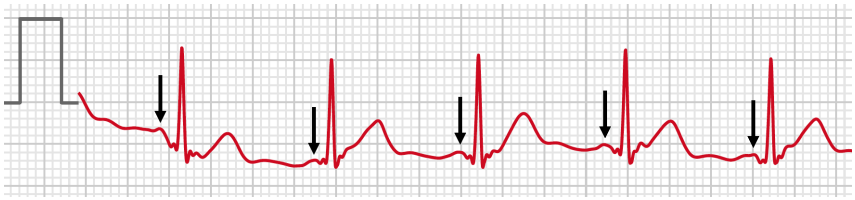
This thesis focuses on analog vector-matrix multiplication on BrainScaleS-2. Routines calibrating the analog circuitry for inference operation will be presented, along with a measurement showing the long-term stability of the calibration. After characterizing the multiplication operation in artificial test cases, hardware-dependent hyperparameters will be investigated using networks classifying the MNIST dataset of handwritten digits [LeCun and Cortes, 1998] as a performance benchmark.

In terms of a more complex application, the efficient analog vector-matrix multiplication will be used to analyze electrocardiogram (ECG) data for signs of atrial fibrillation. We participate in a competition [BMBF, 2019] seeking the most energy-efficient solution that fulfills an accuracy requirement.

Irregularities in the heartbeat have first been examined using an electrocardiogram in 1908. It revealed distinct features for one type of disease, eventually leading to the term atrial fibrillation [Hering, 1908; Rothberger and Winterberg, 1909; Lewis, 1909]. Occurring in about 1.5 % of the population, atrial fibrillation is a relatively common cardiac arrhythmia [Go et al., 2001].

Figure 1.1 shows two ECG traces: The upper trace depicts the normal sinus rhythm. The intervals between two heartbeats (R-R interval) are regular and there is a P wave shortly before the heartbeat (indicated by the black arrows). The lower trace shows atrial fibrillation [Gutierrez and Blanchard, 2016]. Note the much longer interval between the first two heartbeats compared with the latter two. Also, the P wave shortly before the heartbeat is missing, which is visible most clearly before the last heartbeat, where the trace is the least noisy.

The idea of using artificial neural networks for analysis of ECG traces has been around for a long time [Hu et al., 1993]. With the recent advantages in training deep neural net-



(a) Normal ECG trace showing sinus rhythm.

(b) ECG trace showing atrial fibrillation.

Figure 1.1: ECG traces without and with atrial fibrillation. The top trace is an ECG of myself recorded on an Apple watch, which does not show signs of atrial fibrillation. The arrows indicate P waves, shortly before the heartbeats. The bottom trace shows the typical features of atrial fibrillation, irregular heartbeat and missing P waves. It is taken from Gutierrez and Blanchard [2016]. The square wave in the beginning of both traces shows the scaling, it is 1 mV high and 0.2 s long.

works, such as residual blocks [He et al., 2016], many types of diseases can be recognized. Hannun et al. [2019] use a 34-layer convolutional neural network to distinguish 14 labels, including sinus rhythm, atrial fibrillation and a dedicated noise label.

For the aforementioned competition, we will develop our own classifier on BrainScaleS-2. It only has to distinguish sinus rhythm and atrial fibrillation. We focus on using artificial neural networks for the classification. This thesis provides the hardware configuration and calibration for efficient inference with those. However, finding the best network is a complex task and will be covered by a separate thesis.

Concerning ECG, we will evaluate two different solutions utilizing synaptic plasticity mechanisms available on the chip: Firstly, the synapses' correlation sensors, intended for spike timing dependent plasticity (STDP) are used to compare an input ECG signal with regularly spiking neurons. Secondly, short-term synaptic plasticity (STP) is used to translate timing differences in the input ECG signal to amplitude differences. Both of these approaches were implemented and tested on chip.

2 Principles

The HICANN-X chip is the current version of the BrainScaleS-2 system, targeted at emulation of spiking neural networks [Schemmel et al., 2020]. In the current hardware setup, a host computer sends instructions to a field programmable gate array (FPGA) via Ethernet, controlling the HICANN-X chip in real-time. While both the FPGA and the on-chip microprocessor (Plasticity Processing Unit, PPU) could control experiments autonomously, all work presented in this thesis was done from the host computer. The FPGA is crucial for timing-sensitive parts of the program, but only forwards the instructuions to the chip transparently. Responses from chip, such as read data, spikes or ADC samples, are transferred back to the host computer. Software controlling the experiments is written in Python using a hardware abstraction layer, providing configuration containers for various components of the chip [Müller et al., 2020].

The chip, visualized in figure 2.1, features 512 neuron compartments, arranged in two rows, and two synapse matrices. The synapse matrices are again arranged in two halfs, resulting in four quadrants with 128 neuron compartments each. Every neuron compartment is connected to a column of 256 synapses, so the total number of synapses on chip is 131 072. Adjacent neuron compartments can be connected to form logical neurons, but since these connections are not used in the following, I will simply call each compartment a neuron. Neurons are designed to emulate dynamics given in the adaptive exponential integrate-and-fire model [Brette and Gerstner, 2005; Naud et al., 2008]. Synapses are current-based and stimulate the membrane of the neuron in their column either excitatory or inhibitory. Each synapse stores a local 6 bit weight, which determines the output current. Synapses receive their stimuli from synapse drivers. A synapse driver connects to two rows of synapses, which all receive the same signals. These signals consist of an event label and a `dacen` pulse, activating the current digital-to-analog-converter (DAC). The `dacen` pulse thus determines the time interval during which the synaptic current is emitted, allowing dynamics like short-term synaptic plasticity (STP). The labels, 6 bit of size, allow 64 subsets of synapses within each row to forward different inputs to their neurons.

On BrainScaleS-2, biologically inspired networks are emulated in 1000-fold accelerated time. With my thesis focussing on hardware, all times and parameters are given in the chip domain, not as they would appear in biology. While extending the available number of synapses and neurons in a multi-chip setup is planned, this work was done using single-chip cube setups. The first working chip generation, HICANN-X v1, was already available at the start of my thesis, but the next and improved revision HICANN-X v2 only arrived shortly before the time of writing. I will state the used chip version for each measurement.

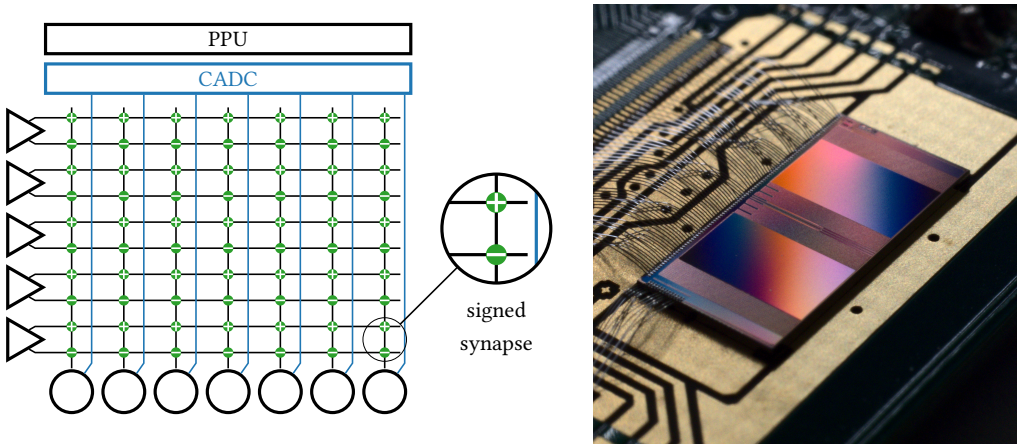


Figure 2.1: Schematic layout (left) and photograph (right) of the HICANN-X chip. The block diagram shows the analog core, including synapse drivers (triangles), neurons (large circles), and synapses (small circles in matrix). Signed weights are achieved by using two synapse rows for positive and negative weights. Figure from Cramer et al. [2020].

2.1 Vector-matrix multiplication

While designed for emulation of spiking networks, BrainScaleS-2 supports vector-matrix multiplication within the synapse array [Weis et al., 2020]. This allows inference with (non-spiking) artificial neural networks within the same substrate. Spiking networks can be seamlessly integrated alongside of, e. g., convolutions. We call this "hagen mode", referring to an older chip designed in our group which was capable of vector-matrix multiplication only [Schemmel et al., 2004].

When operating with STP, synapse drivers store a local state of depression or facilitation for each event label. For hagen mode, the `dacen` pulse width can be modulated using 5 bit of the 6 bit event label. These 5 bit now encode an input strength and are not forwarded to the synapses as label, only one label bit remains to differentiate two sets of synapses. With that, analog multiplication within the synapse array is possible: The 5 bit input activation controls the time Δt to enable synapses, their local 6 bit weight controls the current I reaching the neurons. The emitted charge $Q = I \cdot \Delta t$ is now the product of both parameters. This charge reaches the neuron's synaptic input circuitry and is eventually integrated on the membrane. The membrane potential rises with the injection of charge, a small membrane capitance leads to a steeper dependency. After multiple inputs have been integrated on the membrane capacitor, the potential is digitized using a columnar analog-to-digital converter (CADC). The CADC allows digitization of the membrane potentials of all 256 neurons in a row in parallel.

This mode thus allows analog vector-matrix multiplication. The whole operation is depicted in figure 2.2 for signed synapses, with the upper row of every two rows of synapses stimulating neurons excitatory, the lower row inhibitory. All inputs can stimulate whole

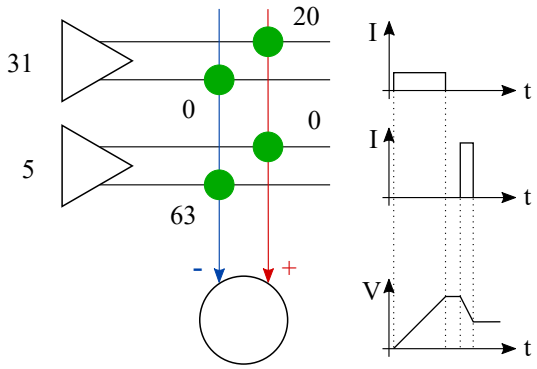


Figure 2.2: Concept of analog multiplication on HICANN-X. Vector inputs, indicated on the left, at the synapse drivers, determine how long synapses are stimulated. The synaptic weight, indicated at each synapse, determines the current. The charge received at the neuron is the product of time and current. Note that the plotted potential on the neuron is simplified, there is actually a time constant introduced by the neurons' synaptic input circuitry. Figure from Weis et al. [2020].

rows of the matrix, the individual multiplication results get accumulated in columns on neurons. From a mathematical perspective the matrix is transposed, since a column vector would be multiplied with a row of the matrix, not a column. The terminology in this thesis will refer to the layout on chip.

2.2 Synapse driver

Synapse drivers are capable of processing STP [Tsodyks and Markram, 1997; Zucker and Regehr, 2002], in an either depressing or facilitating operating mode. Motivated by the release of a finite amount of neurotransmitters, for depression, postsynaptic amplitudes decrease by a configurable utilization for frequent spiking. Between events, amplitudes recover exponentially, with a configurable time constant. Facilitation works the opposite way and lets amplitudes increase with every spike.

Figure 2.3 shows depression and recovery for an excitatory synapse. The black trace, recorded on a neuron's membrane, first contains 10 successive inputs every 5 µs, which lower the amplitude received at every spike. After a pause of 15 µs, a larger spike amplitude is observed as amplitudes recover. The following spikes in light grey were recorded in a similar fashion using the same synapse driver and neuron. There, after the 10 spikes depressing the amplitudes, the wait time before the last spike was increased, waiting 25 µs, 35 µs, ... before the input. The gray amplitudes nicely show the exponential nature of the recovery.

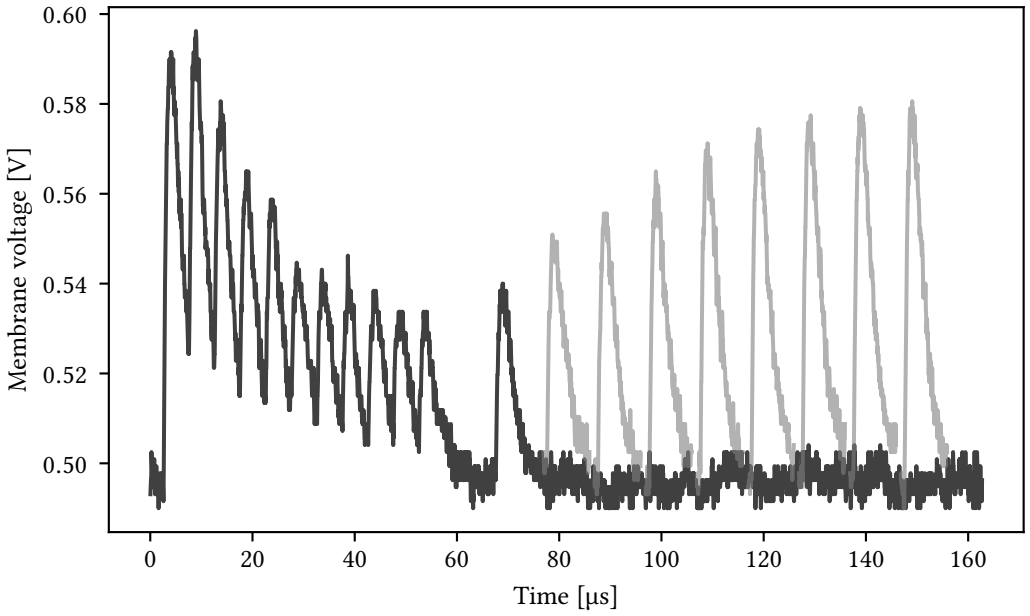


Figure 2.3: Short term depression and recovery of excitatory stimuli on a neuron's membrane. Black trace: The first 10 inputs arrive every 5 µs, depressing the postsynaptic amplitudes. After a longer pause, at a time of 70 µs, the amplitude is larger again, since amplitudes have recovered. Gray peaks: Repeating the experiment with even longer pauses before the final input visualizes the exponential recovery of amplitudes. The traces were recorded using neuron 0 and driver 0 on chip 23 (setup 66, HICANN-X v2).

In hagen mode, the vector entries change amplitudes received at the neurons in the same way. The width of the `dacen` pulse, enabling the synapses, is modulated using the STP circuitry introduced by Billaudelle [2017]. The plasticity state is normally encoded as a voltage on a capacitor, a higher voltage means lower amplitudes. However, this voltage can also be generated by a digital-to-analog converter (DAC), and therefore be controlled externally.

To modulate the `dacen` pulse width, this voltage is compared with a ramp: A voltage, starting low, rises for 2 ns with a current tunable for each driver, allowing calibration of comparator offsets. Afterwards, the ramp rises for 4 ns with constant slope. The `dacen` pulse turns on once the ramp crosses the applied DAC voltage, and is only turned off at the end of the 4 ns pulse generation window. A low voltage output from the DAC therefore means longer stimulation and more charge reaching the neuron. Hence, the desired vector entry needs to be inverted when sent to the synapse driver's DAC.

Each synapse driver stimulates two rows of synapses. While both synapse rows connected to a driver can select excitatory or inhibitory mode individually, they receive the same stimuli if enabled. When using unsigned matrix weights, the one remaining synapse label bit is thus necessary to distinguish inputs for the upper and lower row. Only in

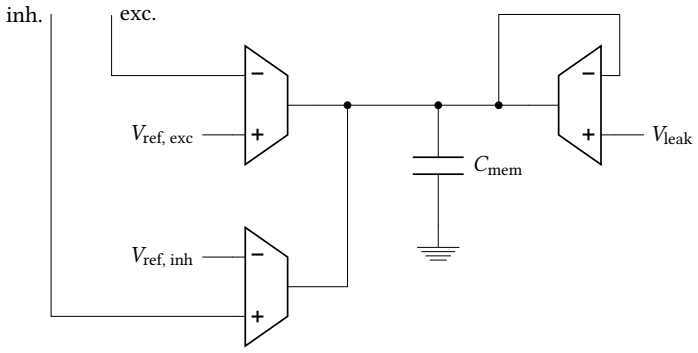


Figure 2.4: Simplified schematic of the analog LIF neuron. On the left, the two synaptic inputs are shown: The lines named "exc." and "inh." arrive from the synapse array, receiving the excitatory and inhibitory synaptic currents. They reach the inputs of operational transconductance amplifiers (OTAs), which generate the current onto the neuron's membrane. On the right, the leak OTA lets the membrane potential decay towards a leak potential.

signed mode, two independent matrices can be placed besides each other, both receiving independent full-size vector inputs utilizing the label bit.

2.3 Neuron

The analog neuron features lots of complex circuitry emulating adaptive exponential dynamics [Aamir et al., 2018; Aamir et al., 2017; Kiene, 2017]. For this thesis, only a subset of features is relevant, implementing the leaky integrate-and-fire (LIF) model. The neuron's membrane is represented as a capacitor. The leak term is implemented as an operational transconductance amplifier (OTA), acting as a resistor pulling the membrane potential to a resting potential. The current-based synapses are realized using another OTA and a resistor controlling the synaptic time constant. Together with the spike threshold comparator, this already completes the LIF neuron. This basic circuitry, shown in figure 2.4, is explained in a little more detail here.

The leak OTA, on the right in figure 2.4, generates a current proportional to the difference between the membrane potential and the leak potential V_{leak}. During the refractory period, it operates in a different setting, allowing a different reset potential. The leak and reset conductivity can also be set individually using two bias current settings. Refractory mode is triggered at every spike, the refractory time is set digitally by a counter.

For matrix multiplication, the neuron is configured to act as an integrator, with little leakage and with the spike threshold disabled. This leaves the synaptic input as the most important component. There are two instances of mostly identical circuitry, shown on the left in figure 2.4. The excitatory input is capable of charging the membrane, the inhibitory input of discharging it.

Each individual synapse, while enabled, sends current from a line connecting all synapses in a column with the neuron to ground. The lines named "exc." and "inh." connect the respective OTA's input to the corresponding synapses. The charge Q "emitted" by a synapse therefore actually decreases the voltage on these lines by an amount $\Delta U = Q/C$, with the capacitance C being mainly the parasitic capacitance of the line. The OTA compares this line's potential with a reference potential and sends a current proportional to the voltage difference onto the neuron's membrane capacitor. The charge reaching the membrane is therefore the time-integral of this OTA's current. The potential on the synapse line is pulled back to the 1.2 V supply voltage using a resistor-like element, not shown in the figure. Tuning this resistance allows setting the synaptic input time constant, with which the synaptic current exponentially decays to zero. Both the time constant and the gain of the OTA affect the amount of charge that accumulates on the membrane.

If the potential on the synaptic line drops below some 1.0 V, the characteristic is no longer linear. Firstly, the current flowing off the synaptic line while synapses are enabled drops, resulting in a smaller synaptic weight. Secondly, the OTA's output current saturates: Further input voltage drops lead to less increase in output current, resulting in a smaller OTA gain. Both these effects lead to amplitudes on the neuron membrane being smaller than desired and will be referred to as saturation of the synaptic input.

There are different strategies mitigating this problem, all having drawbacks. Choosing small synaptic input time constants or lower current outputs per synapse both lead to less signal on the input of the OTA. This decreases signal-to-noise ratio on an already noisy circuit. Small time constants, however, have an advantage over lower currents: After all inputs are sent, the final voltage is reached sooner, while for longer time constants, input currents continue longer. We therefore set the synapses' current output to the maximum value and the time constants to the minimum value. Saturation is now avoided by spacing the input events out over time. This is unfavorable as well, since it results in longer integration times, and even at long membrane time constants, leakage becomes relevant eventually. Generally, since vector-matrix multiplication provides no timing information, maximizing energy efficiency means minimizing execution time. The wait time between events should be tuned for each network layer, with different matrices showing different sensitivity to saturation.

The synaptic input has been redesigned from HICANN-X v1 to v2. The OTA has been changed: It now attenuates noise in the power supply rails better, using a new reference potential generation. Further, the range of possible synaptic input time constants was extended. With noise and saturation of the synaptic input current being major issues, these changes benefit matrix multiplication. Especially saturation is a smaller problem now: The synaptic input time constant could be reduced to 1/4 the value of v1 without sacrificing amplitude. This enables faster integration and mitigates problems with leakage, while noise is still in an acceptable range.

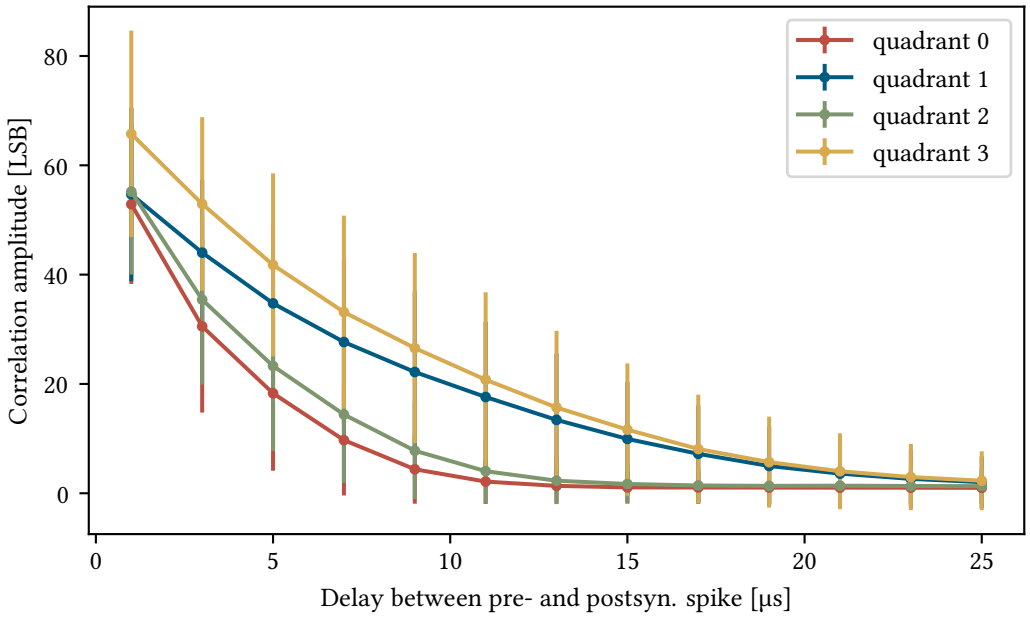


Figure 2.5: Causal correlation amplitudes for different delays between the pre- and postsynaptic events. One trace per quadrant, showing mean and standard deviation of all the synapses on the quadrant. Correlation sensors within the quadrants are calibrated, lowering the standard deviations, but the quadrants' bias currents are not, explaining the differences between the means. Data was measured on chip 09 (setup 69, HICANN-X v1).

2.4 Correlation and CADC

The most widely known learning rule for synaptic plasticity is Hebbian learning [Hebb, 1949], often abbreviated as "what fires together, wires together". In order to determine what fires together, synapses on BrainScaleS-2 are equipped with correlation sensors. Those measure the timing between individual pre- and postsynaptic spikes and accumulate exponentially weighted correlation amplitudes [Friedmann et al., 2017]. These amplitudes can be processed by the on-chip digital microprocessor, the PPU, hence the name Plasticity Processing Unit, enabling on-chip learning. Based on the correlation amplitudes, which are measured with the CADCs' causal and acausal channels, arbitrary rules for weight updates can be implemented.

A commonly used learning rule for spike-timing dependent plasticity (STDP) is a positive weight update for causal correlation, where the presynaptic spike happens before the postsynaptic one, and a negative weight update for acausal correlation, where the spikes occur in reverse. The synapses measure causal and acausal correlation independently. If a prespike arrives, i. e. the synapse forwards an input to the neuron, a causal measurement is started. Once the neuron spikes, this postsynaptic event stops the measurement and changes the voltage on an accumulation capacitor by an amount depending on the time

difference between the two events. The neuron's spike also triggers an acausal measurement, which works similarly and will be stored on a second accumulation capacitor once another presynaptic event arrives.

Correlation amplitudes are digitized using the CADC [Schreiber, 2020], which is capable of recording 512 voltages from one synapse matrix in parallel. Thus, with its two channels per synapse column, it reads acausal and causal correlations from a whole row of synapses. For hagen mode, connecting a readout amplifier from each neuron to these correlation readout lines, the CADC can be used to digitize the neurons' membrane potentials. The correlation sensors were characterized using 20 correlated events with a specific time delta. The accumulated causal amplitudes are shown in figure 2.5. The delay between the pre- and postsynaptic event is plotted on the horizontal axis. Correlation amplitudes depend exponentially on the delay, as expected.

The four plotted traces in figure 2.5 show the mean of all synapses on each quadrant. Error bars indicate the standard deviation of amplitudes within the quadrant. The mean curves are significantly different since the correlation sensors' bias currents, controlling amplitudes and time constants, are not yet calibrated. However, the individual synapses also provide 4 calibration bits, which scale amplitudes and time constants. By simply recording curves for each combination, the settings yielding results closest to the mean of their quadrant were selected and used for the plot. This successfully reduced the deviations within the quadrants.

For digitization, the CADC uses a common voltage ramp for each quadrant and comparators for each channel. The ramp starts at a known low voltage and rises linearly to 1.2 V. In each of the 256 channels per quadrant, an 8 bit counter starts when the ramp voltage starts rising. A comparator stops the counter once the ramp voltage exceeds the input voltage. These counter values now form the digitized results. The counters reach their maximum value of 255 within approximately 1 µs. Hence, the desired dynamic range determines starting voltage and slope of the ramp.

2.5 Training with hardware in the loop

Even on a calibrated chip, not all neurons and synapses work identically. There are fixed pattern deviations left, which will impact performance of networks, if their weights are simply transferred to the chip after training on a CPU or GPU. The training process of artificial neural networks generally involves an inference run and a backpropagation algorithm, which computes weight updates based on the results from inference. In order to compensate fixed-pattern deviations of the hardware, we execute the inference run on HICANN-X, thus including the hardware into the training loop. The weight updates now take the chip's characteristics into account, counteracting the distortions. Our implementation follows the principles of Schmitt et al. [2017] and Cramer et al. [2020], it is described in detail by Spilger et al. [2020].

3 Calibration

BrainScaleS-2 is manufactured in TSMC's 65 nm CMOS process. Like in any other process, tolerances lead to mismatch between identically designed components. In a digital system, a transistor conducting less well typically means some switching process takes more time, resulting in a lower stable clock frequency. For the emulation of neural dynamics, however, time is an important aspect, as most differential equations involve time. As an example, consider a LIF neuron which, due to mismatch, has a shorter membrane time constant than desired: Leakage is enhanced and stacking inputs to reach the spike threshold is harder. Thus, transistor-level variations affect the characteristics of the analog circuits.

To achieve the desired dynamics for all components, there are uniquely tunable parameters for each neuron, 8 voltages and 16 currents. Other circuits, most notably synapse drivers, synapses' correlation sensors and CADCs, also provide calibration mechanisms. During this thesis, we developed software which is capable of calibrating these components, based on the Python API of Müller et al. [2020]. For each neuron, the 10 analog parameters required in the LIF model can be calibrated individually.

For matrix multiplication mode, lacking temporal dynamics, only a subset of the full LIF neuron calibration is required. Calibration mainly targets matching operating ranges of different components. For example, the neurons' membrane voltages need to match the input range of the CADCs, digitizing these potentials. Without calibration of such technical parameters, matrix multiplication would not work. Further, parameters like the strength of synaptic inputs acting on the neurons' membranes are calibrated. Only if inputs follow the same characteristics for all neurons, matrix multiplication works with the same weights as on a CPU. However, when continuing training with hardware in the loop, networks can learn the fixed-pattern deviations. We will later discuss to what extent training with hardware in the loop can replace this part of the calibration.

3.1 Software structure

Due to the multitude of parameters which need to be calibrated, a generic structure was developed, from which single-parameter calibration classes are derived. Generally, a calibration should find a set of parameters which yield results on chip closest to a target result. While parameters, due to their digital nature, are integers, the results can be floating point values, for example fitted time constants. Calibration targets can be a single parameter for all instances (e. g., neurons), or an array with different targets for each instance.

The base calibration provides a `run()` method which uses an algorithm to find optimal parameters. An algorithm can call the calibration's `configure_parameters()` and `measure_results()` functions, which set up an array of parameters on hardware, and

measure their impact on the chip. Algorithms can be very different: they will typically require multiple runs of setting up parameters and checking the results, but could also use a known characteristic and only a single offset-measurement to find optimal parameters. Apart from such algorithms using fitted data, algorithms like a binary search work well as long as parameters affect results in a monotonic way.

The neurons' voltage and currents are generated in a capacitive memory (CapMem) based on 10 bit parameters [Hock et al., 2013]. We observed an issue that arises when setting many CapMem cells to the same digital setting: If shared by a large number of other cells, the generated current or voltage will be different from the value obtained with only a single cell active. To work around this issue, a uniform random noise of ±5 LSB is applied to parameters which would normally be set constant for all neurons. Thus, when calibrating a neuron parameter, the binary search is started with noise instead of all identical parameters. This noise, however, requires the binary search to execute some extra runs in the end, since an instance starting with a lower value would otherwise not be able to reach the top end of the parameter range. Besides the neurons' parameters, the CapMem generates several global voltages and currents. For those, the noisy start of the algorithm is not necessary, as the number of CapMem cells to configure is low.

Since most calibrations involve parameters generated by the CapMem, it is worth to further note that changing the digital settings does not affect the generated voltages or currents immediately. CapMem cells are updated by connecting a small capacitor to a voltage ramp at the right time. Until the desired new potential has been reached at the cell's capacitor, multiple such updates are required. To reach deviations of less than 1 LSB even after big parameter changes, up to 15 updates have to be done [Hock, 2014]. On HICANN-X v1, the ramp runs approximately every 1.5 ms, on v2 it may be even less frequent. Therefore, after configuring a new set of CapMem parameters, a wait of 20 ms is used to achieve a stable state before measuring results. This yields enough precision even for a binary search with its large parameter changes at the beginning. The CapMem provides a special mode targeted at reducing the necessary wait time, which is not used currently.

Based on the generic calibration interface and the common algorithms, calibration classes for each parameter are derived. There were 22 such calibration classes developed, each calibrating only a single parameter. While multi-dimensional calibrations are not supported, dependencies in the form of re-calibrating another parameter before measuring results are possible. For example, when changing an OTA's bias current, its reference potential is re-calibrated before measuring the characteristics with the new bias current. A more complex calibration can use a `prelude` to reconfigure the chip and restore the original state in a `postlude`. Both get called automatically when `run()` is called on an instance of a calibration.

For typical usage, the numerous calibrations of all individual parameters are encapsulated in larger functions. As an example, the hagen-mode neuron calibration takes parameters like the resting potential, membrane and synaptic input time constants and strength of synaptic inputs, and calls all individual parameter calibrations in a suitable order internally. Similar functions wrap the spiking LIF neuron, the CADC and the synapse driver

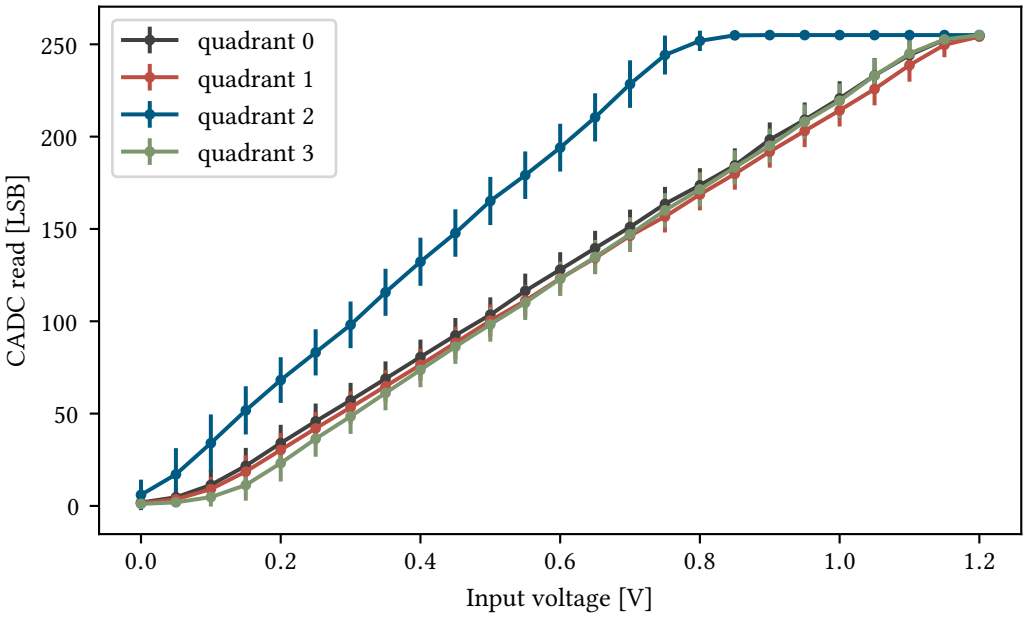


Figure 3.1: Characterization of the CADCs digitizing an externally supplied input voltage while uncalibrated. The mean reads of all channels on a quadrant are plotted, error bars indicate the standard deviation of the 256 channels within the quadrant. The quadrants' results differ since ramps are uncalibrated, also the individual channels show systematic offsets. Data was measured on chip 09 (setup 69, HICANN-X v1).

calibrations. A calibration result object can be used to serialize the obtained parameters and apply a stored calibration later. With vector-matrix multiplication working well for most use cases with only one set of calibration parameters, it is possible to use the same single line of code to calibrate the chip in most experiments, which takes roughly two minutes.

3.2 CADC

For accurate digitization of neurons' membrane potentials or synapses' correlation measurements, the CADCs [Schreiber, 2020] need to be calibrated. Using a voltage ramp and comparators in each channel, both quadrant-global ramp properties and channel-individual comparator offsets need to be calibrated. The dynamic range for digitization can be selected by providing two reference voltages at the lower and upper end. While these are currently produced by the CapMem, using an external DAC with well-known properties is also possible.

The CADC ramp's starting voltage is calibrated near the lower end of the dynamic range for digitization. Setting a typical, yet uncalibrated ramp current, a low digitized read

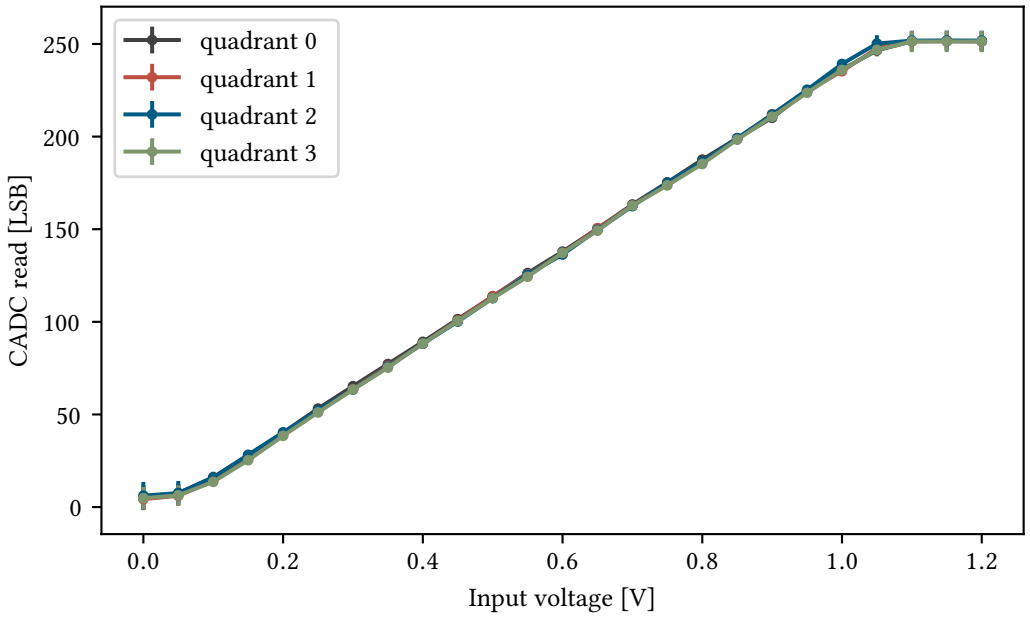


Figure 3.2: Characterization of the CADCs digitizing an externally supplied input voltage in calibrated state. The mean reads of all channels on a quadrant are plotted, error bars indicate the standard deviation of the 256 channels within the quadrant. All four quadrants' characteristics have now been aligned, channel offsets have been calibrated up to a standard deviation below 1 LSB, rendering error bars invisible. Only near the ends of the dynamic range, when channels with higher offsets start clipping, these deviations are larger. Data was measured on chip 09 (setup 69, HICANN-X v1).

is expected with the lower reference voltage applied. To leave headroom for individual channel offsets calibrated later, the expected CADC read there must not be zero, but some small value. For the default dynamic range, a value of 20 LSB works well.

The CADC ramp slope, controlled by a current flowing onto a capacitor, is now calibrated using the reference voltage at the upper end of the dynamic range. The ramp current yielding a high CADC read there, 230 LSB by default, is searched. Again, the expected value must be smaller than the maximum value of 255 LSB to allow for channel offset calibration. Further, the CADC ramp only rises until 1.2 V and steepness already decreases above approximately 1.1 V. This constraints the choice of the upper reference voltage.

After equalization of the ramps, calibrating the individual channel offsets remains. Each channel compares the ramp voltage to its input voltage and stops a counter once the ramp voltage exceeds the input. At the point of triggering, the voltage difference between the inputs of the comparator is non-zero, due to transistor-level mismatch. The digital results are therefore shifted by a certain positive or negative number to compensate these offsets. For calibration, a reference voltage in the middle of the dynamic range is applied,

and all channels' results are recorded. We calculate the median of all reads and configure the deviations from it as offsets for each channel, where they are applied to future reads automatically. Note that these channel offsets result from a constant voltage offset and therefore get larger when using a narrow voltage range, i. e. a small ramp current. In this case, the aforementioned headroom when calibrating the ramp properties may need to be enlarged to avoid early clipping of channels with higher offsets.

Figures 3.1 and 3.2 show the characterization of the CADCs digitizing an external reference voltage, for the uncalibrated and calibrated state, respectively. There, we plot the mean and standard deviation of the channels in each quadrant. The uncalibrated state (figure 3.1) shows that the ramps of the four quadrants are not aligned and deviations between channels are still high. In figure 3.2, calibration has equalized the characteristic across the quadrants and matched a target range of voltages. Also, the error bars are much smaller, with digital channel offsets now applied. The standard deviation between the channels was below 1 LSB of the 8 bit result, which corresponds to approximately 4 mV of input voltage. Here, the dynamic range for digitization was chosen to be roughly 0.1 V to 1.0 V, which could still be extended a bit if necessary. The results look very good and allow using the CADC for most of the remaining calibrations.

3.3 Neuron

Calibrating the LIF neuron circuitry is the most complex part of the presented calibrations. A number of potentials, time constants and amplifier gains need to be set: The potentials include the leak (resting) potential, reset potential (during refractory period) and the spike threshold voltage. Time constants are found in the synaptic inputs (excitatory and inhibitory) and the membrane itself (leak time constant). The strength of the synaptic inputs is set via the gain of an OTA (excitatory and inhibitory).

3.3.1 Potentials

Calibrating leak and reset potential is very straight-forward. The calibration simply reads the resting potential or the potential during reset, finding parameters such that the obtained CADC reads are close to the targets. Calibrations like these take some 1 s of time, using a binary search with noisy start values.

Calibrating the spike threshold involves a bit more logic: With the reset potential low and the leak potential calibrated at the desired spike threshold, the threshold is moved until the neuron spikes slowly, but regularly. The average inter-spike interval is targeted to be many times larger than the refractory period and membrane time constant, ensuring the threshold is close to the leak potential, and not significantly higher like in a typical leak-over-threshold setup with regular spikes.

With the synaptic inputs disabled and therefore only little noise on the membrane, this method works well. There are some systematic offsets to the obtained threshold, and some

variations between the neurons resulting from the method of slow, but regular spiking being very susceptible to noise. For my experiments at lower potentials, the results have been sufficient. However, when setting the spike threshold higher, the mechanism fails: The leak voltage is dropped by a source follower and can thus not reach higher voltages. A different approach using a constant input current, which charges the membrane capacitor until it reaches the spike threshold, then being reset to a calibrated potential could be used at higher thresholds. If the currents, reset potentials and membrane capacitances are equal, the spike rate will be equal too, serving as the observable during calibration. But since the existing method served my spiking experiments well, I did not use this alternative approach yet.

3.3.2 Time constants

In the LIF model, leakage pulls the membrane back to a resting potential. This is achieved using an OTA which behaves like a resistor, connecting the membrane capacitor to the leak potential. The membrane time constant τ_{mem} is given by the quotient of membrane capacitance C_{mem} and leak conductivity g_{leak}: $\tau_{\text{mem}} = C_{\text{mem}}/g_{\text{leak}}$. For calibration of the membrane time constant, we set the membrane capacitance as a fixed parameter and calibrate the leak conductivity, controlled by the OTA's bias current, such that the desired time constant is reached. To do so, two different approaches were explored.

In case of long time constants, larger than 30 µs, a single CADC sample can be used to estimate the time constant. Using this method, the membrane potential is first brought to a known voltage V_{reset} below the leak, using the reset. Releasing the reset, the potential rises exponentially as it decays back to the leak voltage V_{leak}. After the desired membrane time constant, a single CADC sample is recorded. The measured voltage should be $V_{\text{sample}} = V_{\text{leak}} - e^{-1} \cdot (V_{\text{leak}} - V_{\text{reset}})$ since after one time constant, the exponential decay has proceeded to $1/e$ of the original amplitude. If the acquired voltage is higher than this target voltage V_{sample}, leakage happens too fast, otherwise it is too slow. The refractory time needs to be considered when calculating the wait time between membrane reset and CADC read. The advantage of this method is its simplicity, with only a single, directly measured value to be compared to a target, and its parallelism, since neuron resets and CADC reads can happen in parallel.

For shorter time constants, when measuring only one voltage sample, the design of the CADC, using a ramp, becomes an issue. Since there is no sample-and-hold feature, we have to keep in mind the input voltage from the neuron is still changing while the CADC ramp is running, which takes approximately 1 µs. A lower input voltage is sampled earlier in this interval, since the ramp exceeds this low voltage early, than a higher input voltage, only exceeded at the end of the ramp. This limits the timing precision and, considering the exponentially rising voltage, can become an issue for short time constants.

The better method for shorter time constants is using the fast membrane-ADC (MADC), which samples with a frequency of approximately 30 MHz. In contrast to the CADC, it can

only connect to one neuron at a time. After recording the exponential decay of the membrane potential due to leakage, the time constant is fitted to the acquired trace.

The leak and reset potential do not need to be calibrated precisely, as long as they differ significantly enough for the fit to work well. Also, this assumes the membrane capacitance and leak conductivity to be independent of the membrane voltage, which is a fair approximation, but not true generally. It may therefore still be advisable to calibrate leak and reset potentials such that the covered range coincides with the voltage range used during subsequent experiments. This caveat also holds for the CADC-based calibration, but since the precision of this method is lower anyway, using only a single sample, the effect may not be as relevant.

The disadvantage of this method is only its runtime: On hardware, all neurons have to be measured sequentially, which takes 30 ms. This is not much more than the required wait for the CapMem to stabilize its outputs, which is 20 ms. The main runtime penalty is executing the 512 fits to the acquired samples on the host computer, currently taking some 2 s. Implementing multithreading should yield advantages here.

The time constant of the synaptic inputs can be calibrated with the MADC using the same approach. Instead of resetting the membrane potential, a few input events are used to decrease the voltage on the OTA's input. Again, the exponential decay of the voltage back to its resting voltage at 1.2 V is fitted. Here, only the MADC-based calibration was implemented, since time constants above 30 µs are rare.

3.3.3 Synaptic input OTA

Besides the resistor-like element controlling the time constant, the most important part in the synaptic input is the OTA. This amplifier emits a current onto the membrane capacitor, depending on the difference between its two input voltages. One input voltage is the line connecting the synapses, where every stimulation of a synapse decreases the potential. The other input voltage is a reference voltage of approximately 1.2 V, matching the baseline potential on the synaptic line. Mismatch during manufacturing introduces an offset between the two inputs, which requires calibration of the reference potential. The OTA was alredy investigated and calibrated by Leibfried [2018] as included on HICANN-X v1, but redesigned on v2.

At optimal reference potential, the OTA does not produce any current with no inputs from the synapse array, i. e. when the potential on the synaptic line is at its 1.2 V baseline. To measure the current, the membrane potential is used. With the synaptic input disabled, a baseline CADC read at the leak potential is done. After enabling the OTA, the potential should not change as no current should be emitted. If a constant current flowed onto the membrane, the potential would increase until an equal current flows off the membrane via the leak OTA. The resting potential would thus be increased. For reasonable leak conductivities, the change in resting potential can therefore be used as an observable for the OTA's current output.

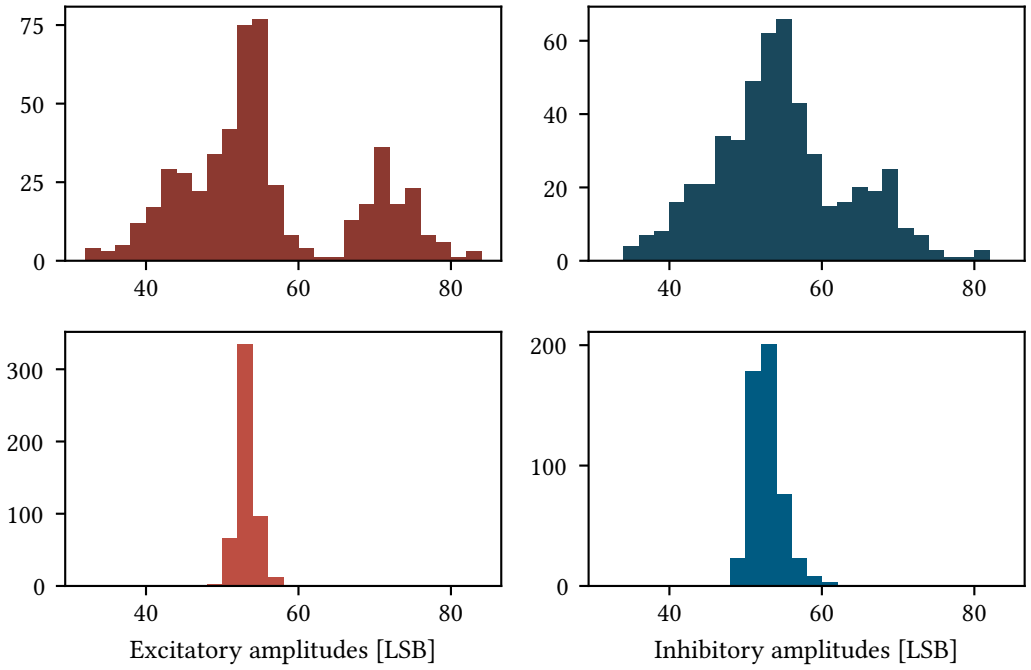


Figure 3.3: Histograms showing the excitatory amplitudes (left, red) and inhibitory amplitudes (right, blue) before (top) and after (bottom) calibration. All neurons receive identical synaptic inputs in all cases. The wide distribution of amplitudes is narrowed down by calibration, using the median of the uncalibrated state as calibration target. All 512 neurons on chip 23 (setup 66, HICANN-X v2) are included in the histograms.

The proportionality factor between the OTA's current output and the voltage difference on its input, its gain, is controlled by a bias current. After changing this bias current, the offset between the two inputs also changes, which requires the reference potential calibration to be repeated. Due to this dependency, calibrating the synaptic input strength requires significantly more time than a usual calibration. For measuring the strength of the synaptic inputs, the CADC is used once again. In a hagen-mode like setup with long membrane time constants, a number of synaptic events are integrated on the membrane capacitor. The resulting potential should be equal on all neurons, the OTA's bias currents are calibrated to reach that state.

The actual calibration target is the mean amplitude observed across all neurons. The number of events used for stimulation needs to be such that the amplitudes are neither too high and in saturation nor too low and therefore more affected by trial-to-trial noise. Starting with 15 events every 1 µs, which is suitable for weaker inputs, tested at one synapse of maximum weight, synaptic input time constants of 1 µs and bias currents of 100 LSB, the number of events is decreased until they are no longer near saturation, in case stronger inputs are configured. If both synaptic inputs, excitatory and inhibitory, shall be equally

strong, after calibrating the excitatory synaptic input in this fashion, the target amplitude and the number of events used for stimulation is reused for the inhibitory input.

When calibrating the input strength in this fashion, all possible differences in inputs to the OTAs are absorbed in the calibration of the strength, calibrating more than the actual gain. For example, the DAC bias current for the synapses, scaling the current output at a given weight and thus scaling the charge reaching the OTA's input per event, differs between quadrants. It is a global bias current per quadrant, but due to differences between each quadrant's CapMem instances, the mean current output differs. It is currently not calibrated since the maximum bias current is desired for the best signal-to-noise ratio anyway, as discussed before. The introduced deviations are absorbed in the OTA's bias current.

Further, the synaptic input time constant affects the amplitudes observed on the neurons, as the OTA's output current onto the membrane decays slower, resulting in more charge output per event. For HICANN-X v1, the minimum synaptic time constant is approximately 1.3 µs, which is large at least for matrix multiplication. Therefore, for hagen mode, the synaptic time constant was not calibrated, but set to its minimum value. The resulting differences were also absorbed in the synaptic input OTA's bias current. For HICANN-X v2, synaptic input time constants can be much lower. Thus, we now calibrate the time constant at a value of 0.32 µs, allowing inputs to be sent four times faster than on v1 without saturation of the synaptic inputs.

The amplitudes of synaptic inputs before and after calibration of the OTA's gain on HICANN-X v2 are shown in figure 3.3. In the top left histogram, uncalibrated excitatory amplitudes are shown. Their distribution is wide, one quadrants' neurons even have separated a bit, with amplitudes mostly larger than 65 LSB. This is probably due to this quadrant's CapMem driving higher currents at the same settings, which means two things: Firstly, the synapses emit higher currents due to the uncalibrated synapse DAC bias current. Secondly, the synaptic input OTA receives a higher bias current, leading to a higher gain. Thus, both effects increase the amplitues received at the neuron's membrane. Uncalibrated inhibitory amplitudes (top right) show a similar distribution, although the mentioned quadrant is not separated as clearly.

After calibration, the distribution of amplitudes is much more narrow (bottom row of histograms in figure 3.3). The standard deviation of neurons' amplitudes before calibration was 20 % for the excitatory and 16 % for the inhibitory synaptic inputs before calibration. This, again, includes both the OTAs' and the synapses' DAC bias mismatch. After calibration, the numbers were brought down to 2.4 % for the excitatory and 3.8 % for the inhibitory synaptic inputs. For all of the four histograms, the mean amplitudes of 30 identical measurements in rapid succession were used, in order to eliminate trial-to-trial variations. Across these 30 measurements, the trial-to-trial variation was 3.9 %, approximately constant in all the four histograms. With fixed-pattern noise reduced to the same order as the statistical variations, the calibration works well.

3.3.4 Hagen-mode neuron

As briefly mentioned before, the neuron calibration for hagen mode can be a bit simpler than calibrating the full LIF neuron. The reset potential is simply calibrated at the leak potential, a spike threshold is not required. The refractory period is chosen sufficiently long for the membrane potential to fully reach the reset potential, clearing any previous state. Membrane time constants are calibrated long to achieve integration, synaptic input time constants short.

Figure 3.4 compares membrane traces of LIF neurons (top) and hagen-mode neurons (bottom). For both subplots, the same five neurons are shown, with neuron 0 plotted black, the others in light gray. The upper panel shows LIF neurons calibrated to a membrane time constant of 6 µs and synaptic input time constants of 1.8 µs. The lower panel shows the same neurons calibrated for hagen mode, with a membrane time constant of 60 µs and synaptic input time constants of only 0.32 µs. For all traces, an inhibitory stimulus arrives at a time of 5 µs, and four excitatory stimuli arrive at a time of 25, 30, 33 and 36 µs. All stimuli were sent to two synapses for twice the amplitudes.

For most of the LIF neurons, the third excitatory input did not trigger a spike, although the threshold was almost reached, and the fourth excitatory input then did so almost immediately. One neuron already spiked at the third input, but the voltage at the spike of this neuron differs only slightly from the voltage reached by the other neurons after the fourth input. After the spike, the membrane potential is reset below the leak potential. The neuron that spiked early is no longer in the refractory phase when the fourth excitatory stimulus arrives and shows its full amplitude, initially the potential is even pulled up by leakage.

For hagen-mode, we notice that apart from slight differences in the baselines, all five shown traces also look very similar, especially comparing the amplitudes of inputs. Here, the arriving inputs produce much sharper edges in the membrane trace due to the small synaptic time constants. These small time constants are also the reason why the amplitudes appear smaller in hagen mode: Even with the OTA's gain set higher, the current decays so fast that its integral, the voltage difference on the membrane, is still lower. In practice, inputs from a vector can obviously be sent much faster, at almost 200 times the rate shown here, verified using an MNIST network. We conclude that calibration of both the LIF and the hagen-mode neuron work as expected.

On HICANN-X v1, noise emitted from the synaptic inputs is a major issue. The amplitude of such random currents emitted by the synaptic input OTAs can be decreased using a higher leak conductivity. Therefore, after calibrating the synaptic inputs at equal membrane time constants, an additional optimization of the leak conductivity is done on HICANN-X v1. The leak conductivity is calibrated based on a noise level calculated from the standard deviation of successive CADC reads. Additionally, for high deviations of the resting potential from the targeted baseline, i. e. in case the membrane potential floats, the leak conductivity is always increased. This results in membrane time constants of roughly 50 µs to 200 µs. The noise levels on v2 appear to be lower, rendering this step unnecessary, thus leaving membrane time constants calibrated to 60 µs.

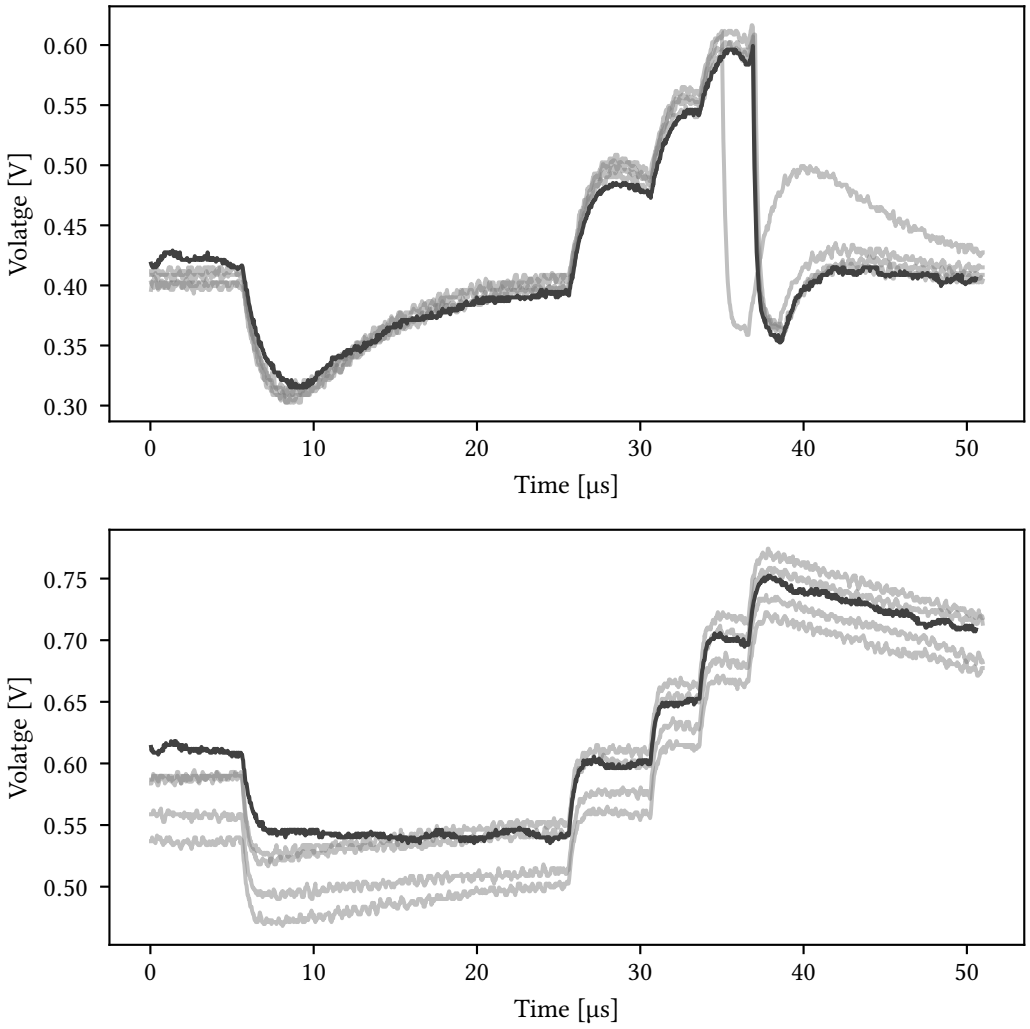


Figure 3.4: Membrane traces for neurons receiving one inhibitory stimulus at a time of 5 µs and four excitatory stimuli starting at 25 µs. **Top**: Neuron 0 (black) calibrated as a LIF neuron, spiking in response to the fourth excitatory stimulus, at approximately 37 ms. Neurons 1-5 (light gray) are calibrated with identical parameters, one spiking already at the third excitatory stimulus. **Bottom**: Neuron 0 (black) calibrated for hagen mode, with a shorter synaptic input time constant and a longer membrane time constant. Neurons 1-5 (light gray) are calibrated identically for hagen mode. Apart from differences in the baselines, amplitudes across the different hagen-mode neurons are very similar. Both spiking and hagen-mode calibration seem to work as expected. The measurement was taken on chip 23 (setup 66, HICANN-X v2).

Another difference between HICANN-X v1 and v2 is the synaptic input time constant. While calibrated on v2, it is set to the minimum achievable value on v1. In order to discuss whether it is reasonable to leave the synaptic input time constant uncalibrated, although it affects the amplitudes observed on the membrane, a short calculation of the charge output is done. The charge Q received per event on the neuron membrane is the time-integral of the OTA's output current $I(t)$. The current starts at an amplitude I_0 and then decays exponentially with a time constant τ_{syn}.

$$\begin{aligned} Q &= \int_0^\infty I(t)dt = \int_0^\infty I_0 e^{-t/\tau_{\text{syn}}} dt \\ &= \left[I_0 \cdot \tau_{\text{syn}} e^{-t/\tau_{\text{syn}}} \right]_{t=0}^{\infty} = I_0 \cdot \tau_{\text{syn}} \end{aligned}$$

For the limit of time reaching infinity, the charge reaching the membrane is indeed the product of the current amplitude per event I_0, controllable by the OTA's bias current, and the synaptic input time constant τ_{syn}. It could thus simply be absorbed in the bias current calibration as well. However, when waiting a finite time, an instance with shorter time constant but higher amplitude has put more charge onto the membrane than a different instance with higher time constant and lower amplitude. The possible wait time is limited by the membrane time constant, since integrated charge decays again. This issue exists similarly concerning the ordering of the events: The first events arriving at the neuron will always have more time to charge the membrane than the last events. We conclude that while it is certainly favorable to have the synaptic input time constant calibrated, it does not introduce new problems, only adding to the already existing trade-off between waiting for synaptic currents and decay of the integrated potential. Importantly, we desire the synaptic input time constant to be short, but can only calibrate it at a greater value on HICANN-X v1, since it has to be accessible for all neurons. Hence, we decided that setting the minimum value suffices on v1, but calibrate the time constants on v2, where we can do so at lower values.

3.4 Synapse driver

The synapse driver's main task is modulating the pulse width that enables its two rows of synapses. Without plasticity enabled, synapses are enabled for 4 ns per event. During this time, synapses discharge the line reaching the neurons' synaptic input OTA. The enabling time of synapses is much shorter than the recharge time constant of this line. The current reaching the membrane is thus accumulated during multiple synaptic events, if they arrive in close succession. The maximum processing rate is one event every 8 ns.

With plasticity enabled, the width of the `dacen` pulse, enabling the synapses, is modulated between 0 ns to 4 ns. A ramp is used to translate a voltage encoding the desired amplitude into the actual pulse. This voltage determining the pulse length is either gen-

erated by analog circuitry processing STP or by a DAC processing hagen-mode vector entries. A comparator starts the `dacen` pulse once the ramp exceeds the applied voltage. Mismatch results in the triggering point of the comparators being shifted by a constant voltage, similar to the mismatch of the CADC channels' comparators.

The ramp can be shifted to compensate these offsets. It is generated by a constant current flowing on a capacitor [Billaudelle, 2017]. Within the 8 ns time interval per event, the ramp is initially reset for 2 ns, and then a constant current flows onto a capacitor for the remaining 6 ns. In the first 2 ns of this ramp generation phase, a configurable DAC can be used to remove current from the capacitor. This means that in the final 4 ns, the ramps can have different "starting" voltages. Only this final time frame is relevant for pulse generation.

The voltage offsets on the STP ramp depend on the current removed by the 4 bit DAC during the precharge phase. The maximum current of the DAC depends on its bias current, but also the available potential at the STP ramp capacitor. If the voltage difference between the capacitor and ground is too small, the DAC removes only little current. Thus, in order to spread the ramp voltages at the beginning of pulse generation sufficiently, we need a certain ramp current, even at maximum DAC bias. The minimum ramp current to cover the calibration range is approximately 600 nA.

Using such ramp currents is not a problem for STP, since the potentials encoding the STP state are between two configurable voltages, representing the fully recovered and fully depressed (or facilitated) state. These voltages cover a suitable range. However, using hagen mode, a DAC needs to generate this voltage. On HICANN-X v1, this DAC has been too weak, its maximum output voltage was much lower than the voltage range covered by the ramp during pulse generation. Using the same ramp current at 600 nA, only pulse lengths of roughly 2 ns to 4 ns could be generated. With times below 2 ns unavailable, we can not map input activations properly. Therefore, the ramp current when using hagen mode was reduced and calibrated such that the ramp voltages concide with the DAC's output range, reaching only values of some 350 nA. For STP, the ramp current is not calibrated, since the two boundary voltages can compensate deviations. For my experiments involving STP, they are currently hand-tuned.

During calibration for hagen mode, estimating the pulse lengths produced by synapse drivers is done using the amplitudes received at the neurons. With CADCs and neurons calibrated, the integrated amplitudes on neurons' membranes are directly proportional to the pulse width emitted by the synapse drivers. A diagonal matrix can be used to record many drivers' outputs in parallel. Note that for lower amplitudes, analog effects from the synapse array can make a difference. For very short pulses, in the order of 100 ps, the edge steepness of the `dacen` pulse may not suffice to properly enable the synapses. To avoid this non-linear end of the characteristic, we use medium pulse lengths during calibration, above 1.5 ns. However, the described issue remains during usage, e. g., when very low vector entries are processed in hagen mode.

In order to calibrate the ramp currents, first a driver with median amplitudes is picked for each quadrant. The ramp current is a quadrant-global parameter and the ramp offsets are yet uncalibrated. Picking the driver with median amplitudes, assuming comparator

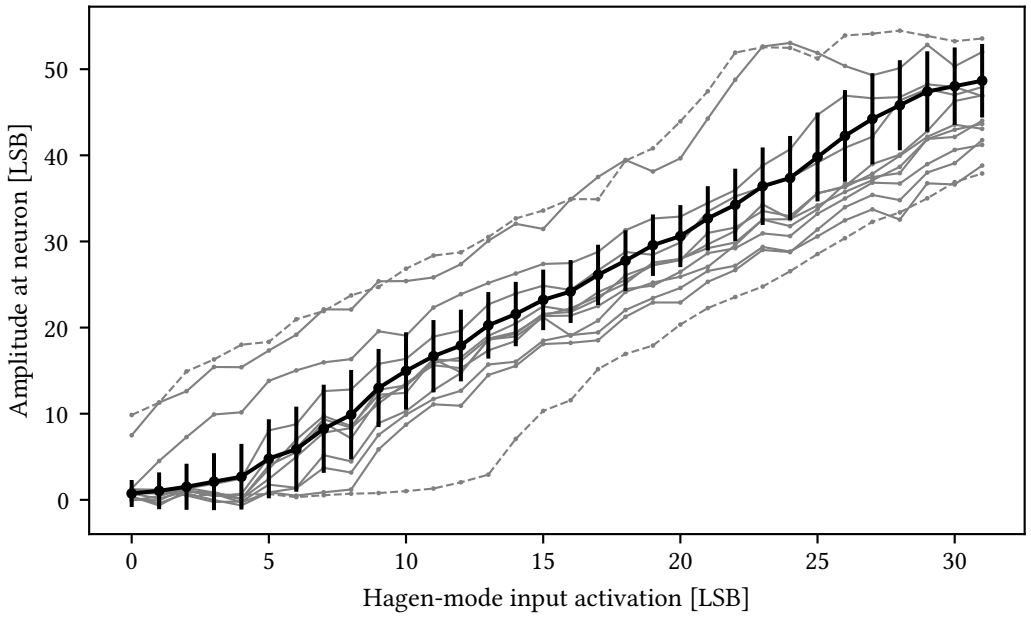


Figure 3.5: Calibrated amplitudes on HICANN-X v1, observed at the neurons for different input activations in hagen mode. In black, the mean and standard deviation across all drivers' amplitudes are plotted. In solid gray, the first 10 drivers are plotted. The dashed lines show the two drivers with the highest offsets. Deviations are large as the available calibration range is insufficient. Data was measured on chip 09 (setup 69, HICANN-X v1).

offsets to be randomly and symmetrically distributed, thus means using a driver with a small comparator offset. For these drivers, the maximum achievable amplitude is measured by using a steep ramp and a low voltage generated by the hagen-mode DAC. Then, the ramp current is calibrated such that a medium voltage generated by the DAC, meaning a medium hagen-mode input activation, results in medium amplitudes at the neuron. While ideally the target amplitude would simply be calculated by the rule of three, an additional factor a little smaller than 1 is used to scale the maximum amplitude down. This factor compensates the fact that the mapping of vector entries to amplitudes received at the neurons is not perfectly linear. While certainly the nonlinearity is partly caused within the synapse driver, the response of synapses receiving very short pulses contributes as well.

On HICANN-X v1, the calibrated ramp currents are much lower than the 600 nA required for good calibration of the comparator offsets. This results in significant differences in the generated pulse widths: The same vector entry yields different amplitudes on different synapse drivers. To help drivers with systematically higher amplitudes reach zero, the ramp currents on v1 have been lowered a bit more, using the scaling factor mentioned before during ramp current calibration. This, however, does not solve the issue. Other drivers now only start generating pulses at higher input activations. Generally, the accu-

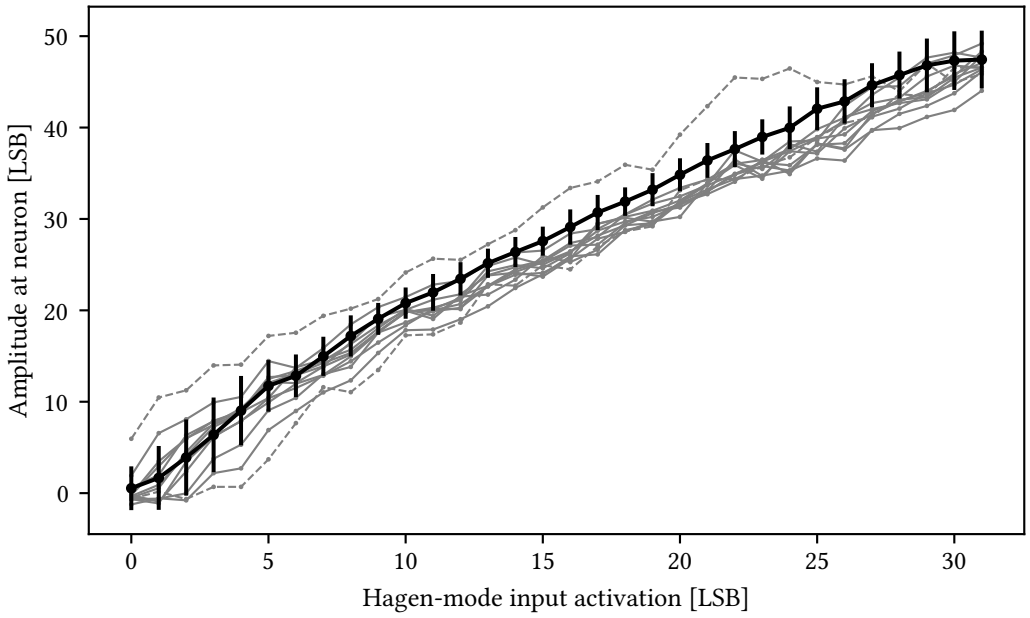


Figure 3.6: Calibrated amplitudes on HICANN-X v2, observed at the neurons for different input activations in hagen mode. In black, the mean and standard deviation across all drivers' amplitudes are plotted. In solid gray, the first 10 drivers are plotted. The dashed lines show the two drivers with the highest offsets. Deviations are much smaller than on v1 since the available calibration parameters have been extended. Data was measured on chip 23 (setup 66, HICANN-X v2).

rate representation of small vector entries has been difficult on v1 due to the insufficient calibration range. The standard deviation of pulse widths was roughly 300 ps after calibration, which is almost 3 LSB in terms of the 5 bit vector entries. The pulse widths were estimated from an amplitude measurement at medium input activation. Note that these additive offsets are a severe problem, they can not be compensated by changing the multiplicative weight matrix. Training can, however, lead to avoiding badly calibrated drivers, or compensate their error at typical inputs. We ensure a vector entry of zero never enables the synapses by not even sending zero-payload events to the drivers.

For HICANN-X v2, the circuitry has been improved. Firstly, the hagen-mode DAC can now generate higher voltages, allowing ramp currents of approximately 550 nA to be used with the same ramp current calibration mechanism. The steeper ramp by itself makes the same voltage offsets affect pulse timings less strongly. Secondly, an additional calibration mechanism has been implemented: The DAC does not only generate a voltage based on the 5 bit vector entries received with each event, but further has a 6 bit offset. This offset covers a largely sufficient range, so large that calibrating the ramp offset using the 4 bit current DAC is no longer necessary for hagen mode. The calibration results are now

much better. The standard deviation of pulse widths at an intermediate input activation was approximately 100 ps, which is roughly 1 LSB in terms of the input activations.

The translation of input vector entries to pulse widths is visualized in figure 3.5 for HICANN-X v1 and 3.6 for HICANN-X v2. Both figures show the 5 bit input activation on the horizontal axis, which is inverted before used as the input for the DAC. The amplitudes received on neurons are shown on the vertical axis. Amplitudes are acquired using 8 neurons, the median of their membrane potentials (mean of the middle two) is used as the synapse driver's characteristic amplitude. This allows recording 32 drivers' amplitudes in parallel while not relying on all neurons working perfectly.

The black line shows the mean and standard deviation of all 256 synapse drivers' amplitudes. The gray lines in the background show the amplitudes of the first 10 drivers, better visualizing typical deviations. The top and bottom dashed gray lines show the minimum and maximum amplitudes across all 256 drivers. It is easily recognizable that calibration works much better on HICANN-X v2. While on v1, the deviations between drivers are huge, they are contained in a much more narrow band for v2. For v2, all of the first 10 drivers plotted in grey seem to yield amplitudes below the mean. This seems not to be an issue with the ramps since these 10 drivers already cover quadrants 0 and 1: every second driver is connected to the same CapMem quadrant, generating the ramp currents. The trend does not continue as systematic as looking the first 10 drivers suggests. In any case, the differences between drivers' amplitudes are small on v2 and should not hinder practical usage of the pulse width generation.

3.5 Long term stability

Beginning in December of 2019, with all calibrations working, the stability of the parameters obtained from calibration was investigated by running them nightly. The parameters may depend on external factors like temperature of the chip, but the laboratory hosting the setups is kept at a controlled temperature of roughly 24 °C. Still, temperatures varied by some 3 K around that mean, with winter holiday marking the minimum and the addition of five powerful host computers in March 2020 leading to an increase in the mean temperature.

Here, again the synaptic input OTAs' bias currents are picked for display, as they are one of the most important parameters. Figure 3.7 shows the excitatory and inhibitory synaptic input bias currents for each neuron over the course of nine months. More precisely, for each neuron, the deviation from the median of its parameter over time is plotted, so the constant spread of parameters across the neurons is not shown. Apart from some single neurons occasionally finding different configurations, the parameters are very stable. Some days, all neurons' results seem to shift up or down by a few LSB, which is probably explained by the target amplitude varying slightly. This conjecture is corroborated by the shifts being identical for both the excitatory and inhibitory bias currents, which are calibrated to the same target. This target, the median amplitude on all neurons before calibration, is measured new every run as well. Over the whole time, the standard

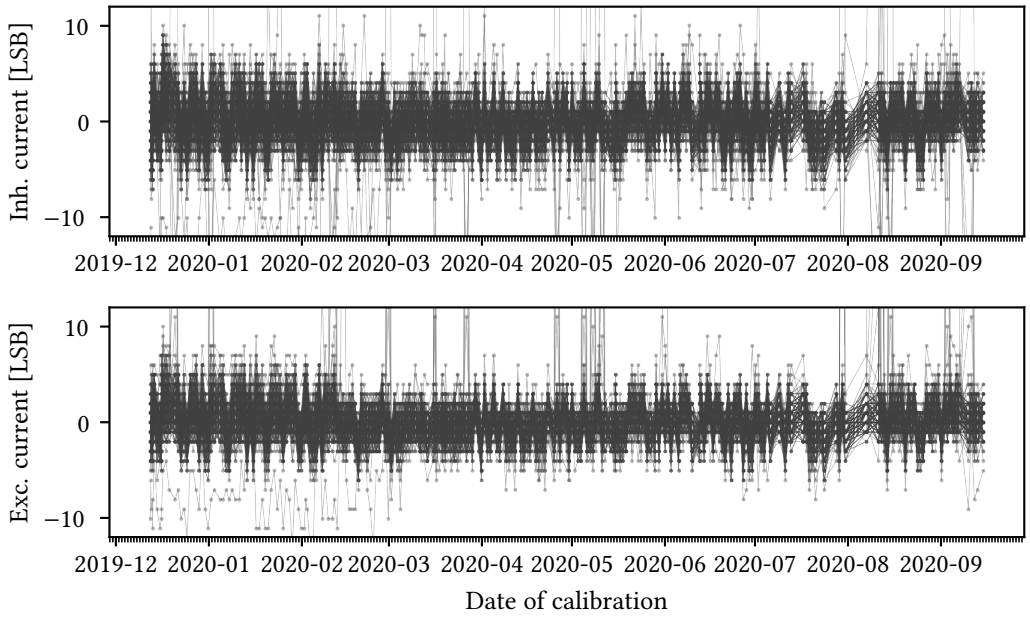


Figure 3.7: Deviations of the individual neurons' synaptic input bias currents from their median over time. Inhibitory (top) and excitatory (bottom) bias currents were recorded in nightly calibrations since end of 2019. The variations of individual neurons are small, apart from some outliers. These are mostly caused by the same neurons. Calibration thus seems to work robustly. The data was acquired on chip 05 (setup 62, HICANN-X v1).

deviation of calibrated parameters is 1.7 LSB in the mean of all neurons. We thus conclude the calibration works reliably and we could reuse old calibration results, which we will later verify using an artificial neural network. Regarding the codebase, tests in continuous integration ensure stability.

4 Hagen mode

For vector-matrix multiplication, the components on the chip are utilized in a certain pattern. Due to the vector inputs not containing any timing information, the instructions to multiply a batch of vectors with a constant matrix are generated in a common way. Certain parameters affecting the analog dynamics are configurable, like the wait time between two vector inputs and the number of vector repetitions during one accumulation phase. This chapter introduces the control flow during a single multiply-accumulate operation, shows the typical problems which arise using the analog circuitry, explains relevant configurable parameters and characterizes the multiplication with an artificial test matrix.

4.1 Executing MAC operations

With the required components calibrated, i. e. the CADCs, neurons and synapse drivers in a suitable operating range, MAC operations follow a constant pattern, as shown in figure 4.1. As a first step, naturally, the synapse weight matrix has to be set to the desired values. In signed mode, positive and negative weights are split into two rows of synapses, which are both connected to the same synapse driver, thus receive equal inputs. After the weight matrix is set, which requires roughly 1.4 ms [Vitali Karasenko, 2020, personal communication], a batch of many vectors can be multiplied with the constant matrix, saving the time for reconfiguration.

Processing one vector of the batch starts with resetting the neurons' membrane potentials to a known starting voltage. This clears any previous state, such as integrated random-walk noise or the result of a previous MAC operation. The potential is reset using the leak OTA, which operates in a high-conductance state during the refractory period. While triggering an artificial refractory state, the membrane potential decays exponentially to a reset potential. This reset potential was calibrated to be equal to the resting potential of the membrane. The time constant of the decay is not calibrated, all conductivities are simply set to the maximum value, yielding time constants much shorter than 1 µs even at maximum membrane capacitance. Hence, we choose a refractory period of approximately 1 µs to allow the voltage to settle at the reset state.

After the resets, the entries in the input vector are sent to the chip one after another. All input activations of zero are skipped, which saves runtime and ensures synapses are actually not stimulated at all. The first entry is sent to the first synapse driver, which stimulates both of its connected synapse rows for a time controlled by the vector entry. In signed mode, one of the rows is configured excitatory, the other inhibitory. Here, we want both rows to react to stimuli, since one of the two weights per column is zero. In unsigned mode, both rows are configured excitatory and should receive independent inputs. Here,

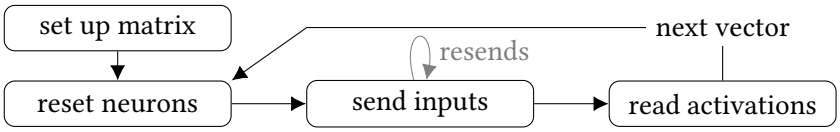


Figure 4.1: Actions on hardware when processing a MAC operation. After setting the weight matrix, the neurons' membrane potentials are reset. Then, the vector entries are sent to the chip, accumulating charge on the neurons. This can be repeated to increase signal. Once accumulation is done, the results are digitized and the next vector can be processed.

the synapse labels need to be used to distinguish the inputs for the rows. With 5 of the 6 synapse label bits used to encode the input activation, the remaining bit suffices to select between two rows. In signed mode, since this selection between rows is unnecessary, the synapse label bit can be used to place two independent matrices besides each other, without sacrificing vector entries. Placement of smaller operations can therefore be more efficient, as not only non-overlapping combinations of smaller matrices can be used to split the synapse matrix.

After all entries are sent, an additional wait is implemented, which allows the synaptic currents to finish charging the neurons' membranes. Then, the CADC is triggered, which digitizes all membrane potentials in a row of neurons in parallel. The result is currently read back to the host computer, but can also be processed by the on-chip microprocessor. With the operation now complete, the next vector can be processed, beginning again with neuron resets. On HICANN-X v1, the synapses' labels need to be rewritten before proceeding, since a bug causes all 6 label bits to be forwarded to the synapses, not only the uppermost one. This means synapses will only react to the inputs if also the 5 label bits encoding the input activation are correct. On v2, it is already verified that this bug is fixed.

4.2 Synaptic input characteristics

The properties of the analog circuitry in the synapse array and the synaptic input stage put some constraints on the design of vector-matrix multiplications. We will first look at the timing during the accumulation phase. For each MAC operation, we provide two parameters to the upper software layers which determine how the vector entries are sent. Firstly, the wait time between two successive events can be configured. Secondly, mitigating noise necessitates repeating the entire vector input within the accumulation phase, increasing the signal and thus the signal-to-noise ratio.

The wait period between two events is an important parameter to configure. After the first event was sent, we wait a certain number of FPGA clock cycles before continuing with sending the next one. Sending an event means sending a spike input packet to the chip, which contains an address selecting the correct synapse driver, and the 6 synapse label

bits, 5 of which encode the desired vector value, as mentioned before. The synapse drivers process the events immediately when received, the timing in this part of the execution is therefore critical. The events then make their way through the synapse matrix and to the neurons' synaptic input stage. The amplitude received at the synaptic input OTA must not be too large to avoid saturation, since both the currents emitted by the synapses and the OTA's characteristic become non-linear if the voltage on the synaptic line is too low. Since the synaptic line is only recharged with the synaptic time constant, a higher wait between the individual events is required in case the average amplitudes are large. For sparser networks, events can be sent faster. However, even the order of the events can matter: If in a sparse network, a few high amplitudes are received successively, saturation will occur, even if the rest of the matrix has low weights. The saturation of amplitudes is discussed in more detail in the following, section 4.2.1.

Once a vector of inputs is sent to the chip, we allow repeating the same vector of input activations. This allows reaching higher signal, which, on a system with large statistical noise, has proven crucial for high accuracy. The noise does not grow as fast as the input amplitudes when using repetitions, resulting in an increased signal-to-noise ratio. We always repeat full vectors, i. e. we start with the first repeated event only after all events were sent once, as opposed to repeating each individual event multiple times. This means the time differences between the unique events of the vector are smaller, thus the impact of leakage distorting the vector is weaker. We expect the conditions on v2 to be better, with less noise and therefore fewer resends necessary. Early results look promising, cutting the number of resends in half is probably possible, but eliminating them might not be. However, we were not yet able to systematically assess the new situation.

After all events are sent, including possible repetitions, the synaptic lines are still below their resting potential and currents are still flowing onto the neurons' membranes. We therefore need to wait some additional synaptic input time constants, in order for the currents to decay to zero. Without this wait, the charge accumulated on the membrane would be too low for the events which arrived last. However, leakage makes the membrane potential decay, decreasing the amplitudes of all previously received events. Waiting too long is thus not an option.

Leakage is already relevant much earlier, starting with the first received event. As shown before in the bottom panel of figure 3.4, amplitudes can decay significantly over the course of some 10 µs. Hence, especially after some resends, the amplitudes from the first inputs are much lower. This, firstly, limits the number of possible resends, since the membrane potential will eventually reach a steady state where the leak current is equal to the input current. Secondly, this means that a new type of "fixed pattern" noise is introduced by the first inputs decaying longer than the last ones. This could easily be compensated by changing the weight matrix accordingly, even without training with hardware in the loop, since it is a deterministic problem, not arising from random mismatch. However, the optimization of the neurons' leak conductivities considering the noise on the membrane, if used, yields different time constants of roughly 50 µs to 200 µs. Further and even more importantly, besides the obvious timing dependence on vector length and wait time between the events, the "pattern" depends on the actual inputs, since we skip entries of zero.

With many zeros in the vector, the first inputs have less time to decay than for a vector filled with more non-zero entries. We thus rely on training with hardware in the loop to compensate the effect for the typical input, but generally we try to keep the time required for one vector input low and the membrane time constant high.

On HICANN-X v1, the minimum synaptic input time constant is approximately 1.3 µs, which is rather long for the intended usage. For a typical MNIST network, saturation limited the event rate to one event every 8 FPGA clock cycles, which, at a clock of 125 MHz, equates to one event every 64 ns. Hence, sending a full vector of 128 entries takes 8.2 µs. This is not very short compared to the membrane time constant, which we typically set to 60 µs at full membrane capacitance. After the last vector entry, we wait another 2 µs for synaptic currents. The first vector input's charge has already decreased 12 % more than the last vector input's one by the time of readout, assuming all charge is injected at the time the event is sent. Utilizing resends, the differences are smaller since the slope of the exponential decay function decreases with time, but they remain, as we do not change the order of the events within a vector.

On HICANN-X v2, the synaptic input time constant is calibrated to 0.32 µs, which is a fourth of the typical value on v1. This allows sending the inputs four times as fast with identical behavior of the synaptic input. With one event every 2 FPGA clock cycles, meaning every 16 ns, sending a full vector takes only 2 µs. The membrane time constants are calibrated to 60 µs. The wait period after the last input can now also be reduced, we're currently working with approximately 0.5 µs. However, since the CADC ramp runs another 1 µs and the potential is only digitized when the CADC ramp crosses the voltage, this timing is not precise. The additional time spent before digitization is the same as on v1, but there, its relative impact is smaller as we wait longer previously. Assuming 1 µs delay between the last event and digitization, the amplitudes of the first vector input have decreased by 3.2 % more than the last inputs. This is in the same order of magnitude as the fixed pattern noise between the calibrated neurons' amplitudes themselves.

4.2.1 Saturation of synaptic inputs

As briefly mentioned before, the neurons' synaptic input stages behave non-linear if too many inputs at higher weights arrive within a short period of time. The effects of this saturation on the neurons' membranes are investigated in more detail here. We generally want to process a vector as fast as possible, however, the synaptic input time constant limits the input rate.

In order to investigate the saturation, the time between two events is swept from 1 FPGA clock cycle, which is 8 ns, the maximum rate without packing of multiple events, to 50 FPGA clock cycles, which is 400 ns, roughly a third of the synaptic input time constant on HICANN-X v1. We generally expect amplitudes to drop when the time between two events is too short, since the synaptic input OTA's output current is smaller than it should be with the amount of events arriving. Note that non-linearities in the OTA's characteristic start at much lower input voltage differences than an actual maximum current is reached.

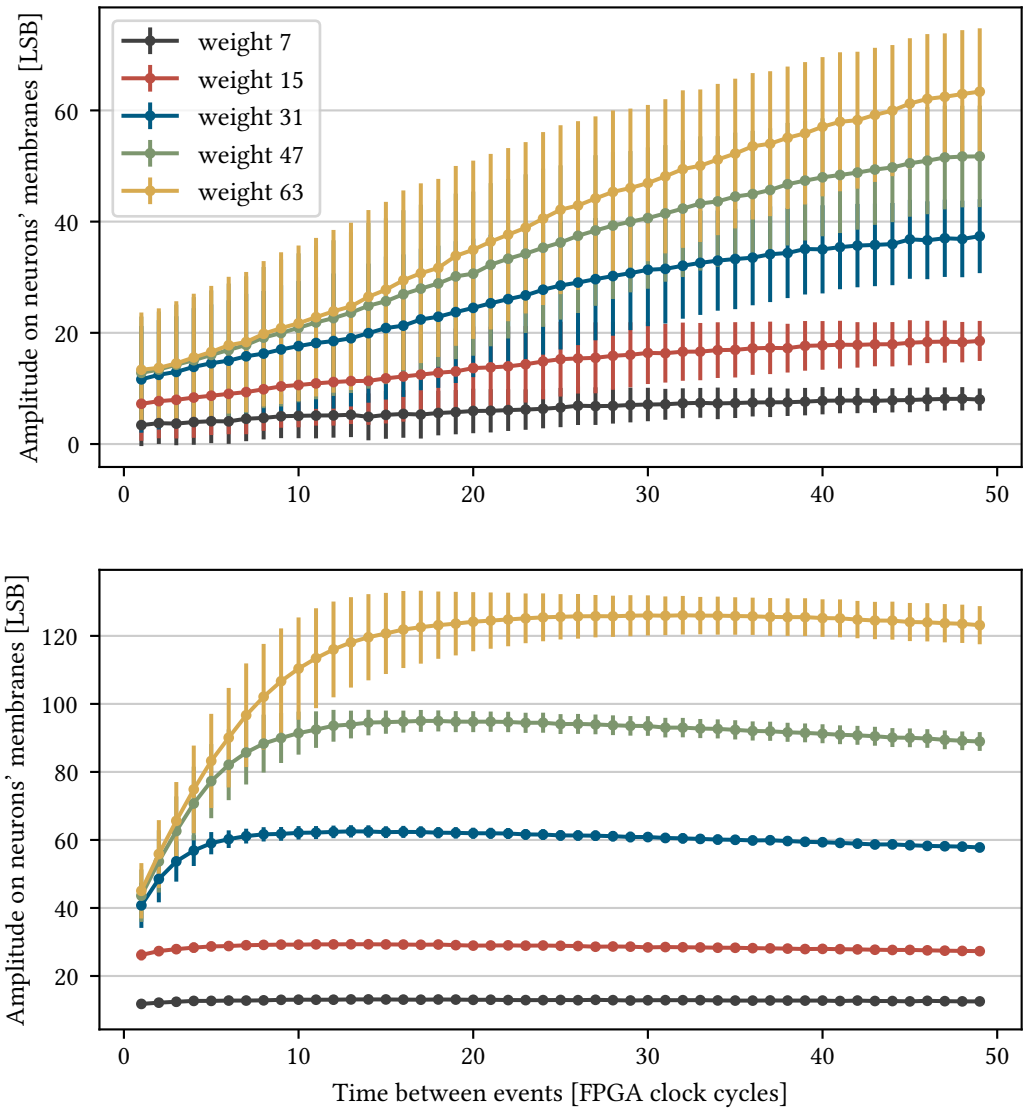


Figure 4.2: Amplitudes received at the neurons for 32 input events at maximum input vector activation and given matrix weight. The time between two successive events is swept, characterizing saturation of the synaptic input. Without saturation, we expect constant amplitudes, independent of the time between them. **Top**: Measurement on HICANN-X v1 (chip 09, setup 69). With the events in rapid succession, non-linearity is large since amplitudes increase with the wait time. For lower weights, amplitudes approach a constant value at higher waits, but higher weights are still not represented linearly when waiting 50 FPGA clock cycles between two events. **Bottom**: Measurement on HICANN-X v2 (chip 23, setup 66). For high weights and little time between the events, saturation is also observed on v2. Amplitudes for lower weights appear constant upwards of only a few FPGA clock cycles time, however. For higher weights, a maximum is observable, since leakage makes the amplitudes decay slightly when waiting longer. This indicates the synaptic inputs now behave linearly. One FPGA clock cycle equals 8 ns.

On the other hand, when waiting too long between two events, leakage of the membrane capacitor makes the amplitudes decay, as described above.

For this experiment, 32 events at maximum input activation are sent to the chip at the given rate, without being repeated. Using only 32 events avoids reaching a high membrane potential, where the digitized value clips, but reaching similar saturation at the synaptic inputs as with a larger vector due to the event rate. The synapses' weights are set to a constant value, multiple runs at different weights are done. For a lower weight, we expect the maximum event rate to be higher, since each event adds less amplitude. For higher weights, staying in a linear operating range requires longer waits between the events.

The amplitudes recorded on the neurons are plotted over the time between two successive events in figure 4.2, the top plot shows the measurement for HICANN-X v1. The lines show the mean of all amplitudes on the neurons' membranes at a certain matrix weight, indicated by the legend. The error bars show the standard deviation of the amplitudes between all neurons. For each neurons' amplitude, the mean of 30 individual measurements is used. The individual neurons' standard deviation across the 30 runs is not shown in the plot, it was approximately 2 LSB on average. The shape of the plotted amplitudes indeed shows small amplitudes on the left side, with the time between two events too low, and higher amplitudes further on the right, when the synaptic input starts behaving more linearly. It seems that at higher weights, even 50 FPGA clock cycles of wait do not suffice to prevent saturation. When using very short intervals between the events, the current onto the neurons' membranes is not only non-linear but reaches a maximum, indicated by the close proximity of the top three lines at drastically different weights. However, one needs to keep in mind that this is a highly artificial test case, and most networks during inference will use lower average weights. Especially using signed inputs would distribute the load on two synaptic input stages. Still, this plot shows that the timing of events needs to be carefully set.

For HICANN-X v2, a similar plot is shown in the bottom part of figure 4.2. First, we notice that amplitudes are generally much higher, even at the left end with high saturation. This is due to the redesigned OTA, emitting less noise and therefore now configured to a higher gain. Thus, the maximum current is higher, and the membrane is charged up more during the limited integration time. Second, we notice that error bars, i. e. the standard deviation of amplitudes across the neurons, is much lower. This is at least partly caused by the now calibrated synaptic input time constants, which make saturation happen equally for the different neurons, while on v1, all neurons are set to their minimum synaptic time constants. For the highest weights, amplitudes are close to clipping on the CADC range, which distorts the result. For the lower weights however, the plot looks beautiful: even at medium weights of 31, waiting times of 10 FPGA clock cycles suffice to avoid non-linear behavior.

Also, the amplitudes on v2 reach a maximum once saturation is avoided and decrease again for higher wait times, while on v1 even traces at lower weights seem to still increase. This is due to the time period between arrival of the first events and the CADC readout, which increases with the wait times. As calculated above, the exponential decay due to leakage can let approximately 12 % of the first amplitudes disappear over the course of 128

vector entries very 8 FPGA clock cycles. Since these plots show wait times of up to 50 FPGA clock cycles, sending the 32 events used here takes up to 13 µs. This reduces the amplitudes of the events which arrived first by nearly 20 %, using a membrane time constant of 60 µs, and neglecting any additional waits between the last event and readout. Following this train of thought, we expect the total amplitude across all 32 events to decrease by 10 %. This is close to what we observe: The amplitudes decrease by 8 % between the maximum and the right end of the trace, measured at a weight of 31 on v2. Since the maximum amplitudes are not reached at zero wait and, further, some delay between the last event and digitization lowers this effect (the lately arrived events decay faster than the early ones), the measurement fits our expectations well. On v1, this effect seems to be entirely suppressed by the charge emitted from the synaptic input stage still increasing with the time between two events.

Reaching reasonable amplitudes and avoiding non-linearities has thus become much easier on v2. In fact, the results look promising regarding our goal of reaching processing times of 5 µs per vector [Weis et al., 2020]: With the current settings, extrapolating from the trace at weight 31, a time of 5 FPGA clock cycles between events looks like a sweet-spot between little saturation and short wait times. With amplitudes there reaching 60 LSB already, using 64 events instead of the 32 would push the digitized membrane potential close to the limits of the selected CADC range already. Sending these 64 events every 40 ns takes 2.5 µs, allowing sufficiently charging the membrane, which is even configured to its maximum capacitance. Larger vectors with lower average weights could be sent faster, this time limit imposed by saturation of the synaptic input can therefore be considered to be constant, at least for homogeneous weight matrices. Adding 1 µs for neuron resets and 1.5 µs for waiting and digitization yields exactly the target of 5 µs. However, it is still to be shown that the used settings are useful defaults and more networks, not only MNIST, work well. This calculation should only serve as an example, not an actual proof: Using signed weights, the inhibitory currents will decrease the amplitudes, depending on the actual weights. Using unsigned weights, the CADC range would typically be extended by selecting a lower reset potential for the neurons. In both cases, more time would be required to "fully charge" the membrane capacitor. Generally, for reaching good signal-to-noise ratios, the wait time between events and possibly vector resends are parameters that will need tuning for each network layer, even on v2.

Another parameter when optimizing the amplitudes at the neurons' membranes is their membrane capacitance. Identical charge output from the synaptic input OTAs means a higher voltage increase on the membrane when the capacitance is lower. However, the noise emitted from the OTAs also increases in amplitude, at least for frequencies higher than the membrane time constant, which would be integrated on the capacitance. Very high frequencies, much greater than 1 MHz, get smoothed out by the readout amplifier not covering the entire spectrum. With the capacitance, the membrane time constant changes, given the leak OTA's bias current is not adjusted. This means lower noise frequencies are compensated by leakage quicker, as the membrane potential would reach a steady state earlier. During the integration phase, however, these random currents would not yet equalize, since integration times generally need to be short with respect to the

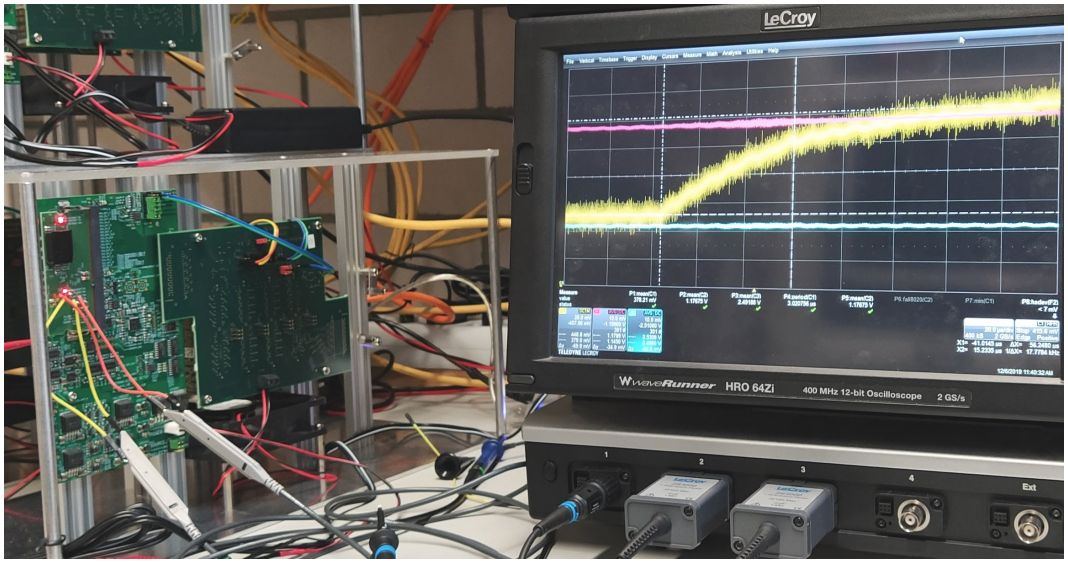


Figure 4.3: Measurement setup showing a transition between two stable resting potentials on a neuron. The left part of the image shows the cube setup with two LeCroy ZS 1000 active probes connected to the power supply rails. On the oscilloscope screen on the right, the yellow trace is the membrane potential, first stable at a lower potential, then increasing to a higher stable potential with an exponential decay. The membrane potential increases by 69 mV with the membrane time constant. The voltages at the power supplies, in red and blue, do not show distortions at the time the transition starts.

membrane time constant. This means that fewer vector sends are possible in order to not lose too much amplitude to leakage. In contrast, the vector entries will lead to higher amplitudes per event. We thus expect a dependency between vector resends and membrane capacitance, but the effect on signal-to-noise ratio is not entirely clear. It could be weak in terms of directly affecting the noise amplitudes, but the reduced integration time could mean there is less time to accumulate noise after the membrane potentials are reset. Assuming the integration of noise is a random walk process, we would expect its standard deviation to grow with the square root of integration time. The effects of all three parameters presented in this section, i. e. the wait time between events, the number of resends and the membrane capacitance, will be investigated using a simple MNIST network later, in section 5.3.

4.2.2 Baseline shifts

On HICANN-X v1, many neurons have more than one stable resting potential, i. e. the membrane potential jumps back and forth between two potentials. The transitions between the two stable states can be rare for some neurons, as seldom as once per minute, while other neurons show these baseline shifts multiple times a second. There could be

neurons with even lower transition rates, which I did not notice during my 5 min of measuring the resting potential.

Investigating the issue, I connected a LeCroy WaveRunner HRO 64Zi oscilloscope to the analog readout pads of the chip, where the neuron membrane was connected. At the same time, I measured the voltage on the 1.2 V and 2.5 V power supply rails, in order to detect possible distortions there, which might trigger the baseline transition. A photograph of the setup is shown in figure 4.3, where on the oscilloscope screen, the baseline transition can be seen: The membrane trace, in yellow, is initially stable at a lower potential, and then exponentially decays to a higher potential. The voltage difference between the lower and upper potential is 69 mV, estimated using the oscilloscope's cursors. The red and blue traces show the 1.2 V and 2.5 V power rails, respectively. Both show the mean of 301 baseline jumps, since the oscilloscope was set to trigger on the positive edge of the membrane potential. At the time the shift begins, both voltages are stable, there are no changes visible.

The time constant of the exponential decay is the membrane time constant, which was verified by changing the membrane time constant on chip and observing similar changes on the oscilloscope. We thus expect that a constant current must start flowing onto the membrane once the potential starts rising. Due to the low leak conductivity used for hagen mode, even a small constant current will lead to this increase in potential: The membrane capacitance is approximately 2.2 pF, the time constant approximately 60 µs, resulting in a leak conductivity of 37 nS. This means that the potential difference of 69 mV can be caused by a current of roughly 2.5 nA.

Disabling the synaptic inputs, i. e. setting the bias current for the OTAs to zero, makes the effect vanish. Disabling only one input, e. g. disabling the excitatory input and keeping the inhibitory input active, makes the effect vanish for some, but not all neurons. Hence, we conclude that both synaptic input OTAs, independently, have a certain probability of producing a constant current. It is not entirely clear how the effect is caused, since all activity on the chip was halted for these measurements, stopping clocks and even turning off the CapMem ramp. The oscilloscope was used instead of the on-chip ADC to further reduce activity. With nothing visible on the power supply rails, and noise sources on chip reduced to a minimum, the transitions were still visible. Since the synaptic input OTAs were already redesigned for HICANN-X v2 anyway, we did not investigate much further. The new situation on v2 has not been assessed in detail yet. While plotting membrane traces over time did show some variations, they were less obviously between two stable potentials. They could be just random low-frequency noise. As on v1, nearly all noise vanishes when the synaptic inputs are turned off.

For the usage of v1, the constant currents mean that the resting potential can not be considered constant after calibration. Instead, we do a baseline read before multiplying a batch of vectors. There, we use a similar pattern to the actual multiplication: After resetting the neuron potentials, we wait some time, roughly the time a vector of input activations would need to be sent to the chip. This means we take a potentially different resting potential into account, which will be slowly reached again after the resets. With the frequency of the baseline transitions being low, in the order of at most 10 Hz, the

baseline read is still valid for some multiplications, which, on v1, take approximately 5 ms per vector. This one baseline read per batch of vectors seems to suffice. For the early experiments on v2, the baseline reads are still in place.

4.2.3 Dependency of synaptic currents on position

For the last section about unexpected behavior of the amplitudes received on neurons, we leave the neurons' synaptic input circuitry and consider an effect within the synapse array. Along a row of synapses configured to the same weight, the emitted currents vary, directly impacting the amplitudes on neurons. The amount of variation with the position of the synapse depends both on the weights and the received input pulse length. This effect is present on HICANN-X v1 and v2.

For systematic investigation, the voltage drop on the line before the synaptic input OTA is measured. After stimulation, the exponential decay back to the idle potential at 1.2 V is fitted using a voltage trace recorded by the MADC. A similar analysis is done during the calibration of the synaptic input time constants, using the buffered readout from the neuron. With the synaptic input time constant calibrated, the amplitude parameter of the fitted exponential decay is plotted for the different neurons. In terms of involved circuitry, a newly added source follower on v2, dropping the synaptic line's potential before the OTA's input and the neuron's readout buffer, is indeed part of this measurement, but otherwise the neuron is unused. All measurements were done on the excitatory synaptic input.

The voltage drops on the synaptic lines are plotted in four subplots, representing the quadrants of the chip. The synapse drivers are located between the left and right halfs of the synapse matrix, the neurons are located in two rows in the horizontal gap between the top and bottom synapse matrix. The neurons are plotted as placed on chip, with neuron 0 starting near the synapse drivers in the top left quadrant, neuron 127 being the furthest away from the synapse drivers in the top left quadrant, and continuing like this through the top right, bottom left and bottom right quadrant.

Using one row of synapses at maximum weight for stimulation, the voltage drop after sending three events as fast as possible, i. e. every 8 ns, is plotted in figure 4.4. The plot shows means and standard deviations of 20 identical experiment runs. We observe that for all quadrants, the synapses close to the synapse drivers emit most charge, dropping the potential on the synaptic line the most. The voltage drop decreases when approaching the outer edges of the synapse array. The magnitude is quite large, the relative deviations reach up to roughly 20 %, but this value varies strongly between the quadrants, with the bottom left quadrant affected the least for this chip. The systematic differences between the quadrants' results are due to the uncalibrated synapse DAC bias current, thus synapses on different quadrants emit different currents at maximum weight.

The effect could easily be compensated by tuning the neurons' synaptic input OTA gains accordingly. However, this dependency of emitted charge on synapse placement gets weaker when reducing the weights, and also changes with the pulse length emitted by the

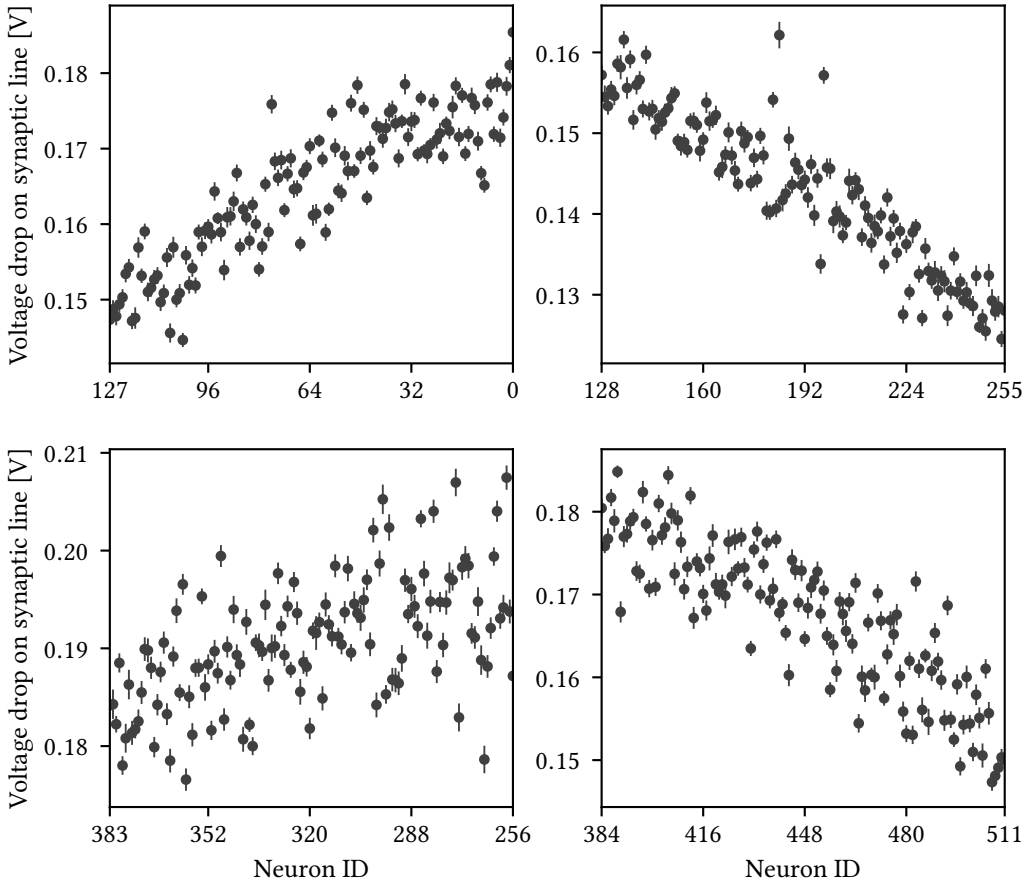


Figure 4.4: Voltage drop on the synaptic input lines for three events at maximum weight, recorded on different neurons. The four subplots represent the hardware layout in four quadrants, data points show the mean of 20 identical runs, the error bars show their standard deviation. The charge emitted by synapses close to the synapse drivers, located horizontally centered between the left and right quadrants, is highest, leading to a higher voltage drop on the center neurons' synaptic input lines. The potential drop on the synaptic lines directly affects the amplitudes received on the neurons' membranes. Ideally, we would expect equal voltage drops for all neurons within a quadrant. The observed dependency of emitted charge on the synapse position varies with both the synaptic weights within the row of synapses and the time the synapses are enabled for. This means the effect can generally not be compensated by calibration. The measurement was taken on chip 23 (setup 66, HICANN-X v2).

synapse drivers. For the presented plot, pulse width modulation was disabled, resulting in the synapses being enabled for 4 ns for each of the three events. Using 8 rows of synapses at a weight of 8 instead of the one row at maximum weight of 63, the effect is barely visible, the relative deviations are reduced to roughly 3 %. Most matrices will not contain full rows of synapses at maximum weight, thus the problem will be much less severe for most networks. However, this is an effect which needs to be accounted for during calibration: If one row at high weight was used to stimulate the synaptic inputs, this effect would be compensated with the OTA's gains, and later re-appear at lower weights, with the opposite dependency. Therefore, the calibration uses eight rows of synapses at lower weight.

4.3 Characterization of the multiply-accumulate operation

Before using hagen mode for actutal inference, its performance is characterized using artificial test matrices. After starting with constant weights to investigate noise, the linearity in both elements of the product is shown using linearly increasing weights and different constant vectors. Finally, a random matrix uniformly drawn from the signed weight range is used to show addition of positive and negative inputs, comparing the hardware results with the correct results computed on CPU.

4.3.1 Constant matrix

As first experiment in hagen mode, a constant weight matrix is used and multiplied with different constant vectors. This allows characterizing fixed-pattern and statistical noise in columns. The differences between synapse drivers are not visible since all 128 drivers' outputs are added up. Repeating the measurement 30 times gives an estimate of the trial-to-trial variations.

This experiment also allows measuring the gain of the MAC operation, i. e. the proportionality factor between the exact result when multiplying the vector entries with matrix weights and the actual result recorded on the CADC. This gain, however, is not constant for our system, but depends on a lot of factors, including the synaptic input OTA's gain, the synaptic time constant, the membrane capacitance, the dynamic range of the CADCs, the synapse DAC bias current, and, in case integration takes longer, even the leak conductivity. While we provide a default set of these parameters and, at least at the moment, change only a few of them between networks, they all vary between chips. The numbers stated below should thus only indicate an order of magnitude. In cases where knowing the gain is relevant, it is measured in a similar way.

Figure 4.5 shows all 512 neurons' response to identical stimuli. The mean and standard deviation across 30 runs is plotted in four grids, representing the four quadrants of the chip. The matrix weight was 12 for all synapses, the vectors consisted of 128 constant entries, covering the full range, the exact values are given in the caption. The wait time between events was set to 8 FPGA clock cycles, the vector was sent only once. For this measurement

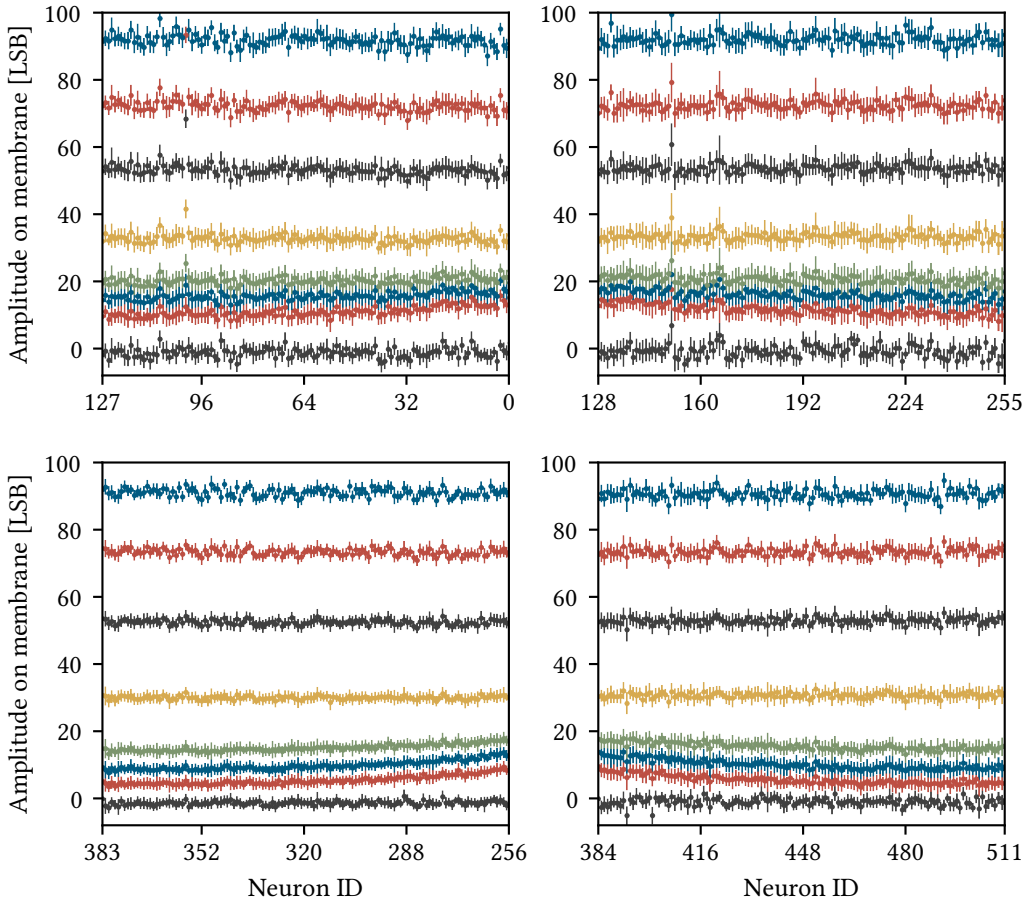


Figure 4.5: Amplitudes recorded on neurons' membranes using different vector inputs at a constant weight matrix. The placement of neurons in the four subplots represents their placement on chip in four quadrants. From bottom to top, the vector entries used for stimulation were 0, 1, 2, 3, 7, 15, 23, 31. The data points show mean and standard deviation across 30 runs. For low vector entries, i. e. low pulse widths, we again observe higher amplitudes close to the synapse drivers. Apart from that, the plot looks as expected, with linearity in encoding the vector entries not ideal, as shown before in figure 3.6. The measurement was taken on chip 23 (setup 66, HICANN-X v2).

taken on HICANN-X v2, the results look very usable: The gain of the multiplication is 0.0019, meaning a single multiplication of $31 \cdot 63$ would result in an amplitude of 3.7 LSB at the neurons. The fixed pattern noise is 2.2 %, the statistical noise is 2.8 %. All these values were calculated at the maximum vector value of 31.

Looking at the figure more closely, we notice that the upper synapse array seems to yield a bit more trial-to-trial variation than the lower two quadrants. While the amplitudes between the neurons and also between the quadrants are generally matched well, they differ near the lower end. The three lowest traces (above the zero-vector) at vector entries 1, 2 and 3 are too high in the top quadrants, with the gap between zero and one large, they are better aligned in the bottom synapse matrix. Still there, the increments at the lower vector entries are higher than in the upper range, most notably the yellow trace at an amplitude of roughly 30 LSB is reached with a vector entry of only 7. Incrementing the vector value in steps of 8 produces the next-higher traces, which are not 30 LSB apart but closer for higher vector values. We observe the not exactly linear mapping of vector entries to amplitudes, which was already shown in figure 3.6.

Further, the lowest three vector entries show again a dependency of amplitude on position, as shown previously in figure 4.4. For these short pulses, this systematic fixed-pattern noise is stronger. The effect vanishes gradually and becomes invisible at vector values of 7 and above. However, this is at matrix weights of only 12 - it might affect even longer pulses at higher weights.

4.3.2 Linear matrix

Since the characterization at constant weight shows that all neurons are working and the vast majority of them is behaving as expected, the linearity of the multiplication is characterized. Now, the synapse weight matrix is set to increase linearly from left to right: Starting with a column of all -63 on neuron 0, the weight increases by one per column, reaching weight 0 at neuron 63, and weight 63 at neuron 126. The weights in column i are set to $w = i - 63$ if $i < 127$. The weights in the synapse column of neuron 127 are all set to zero. For neurons 128 to 255, the remaining synapse matrix is filled with random weights, drawn uniformly between -63 and 63. We thus expect amplitudes close to zero for these neurons, since the expectancy value of each weight is zero, and we add 128 of them. For ease of depiction, the linear and random parts are split into two plots, although they were measured at the same time.

The top part of figure 4.6 shows the top left quadrant of a HICANN-X v2 chip configured to linearly increasing weights. The synaptic weights affect the results linearly, as expected. For the right part of the figure, also the vector entries seem to affect amplitudes nicely, with a vector entry of 1 yielding amplitudes just below a third of those at vector value 3. Also, vector value 7 is at roughly twice the amplitude of value 3, so only sightly too weak. However, on the left side of the plot, at minimum matrix weight, the neurons close to the synapse drivers were used - and again, the absolute value of the amplitudes at a vector entry of 1 is much higher there than at the outer edge of the array, which is on the

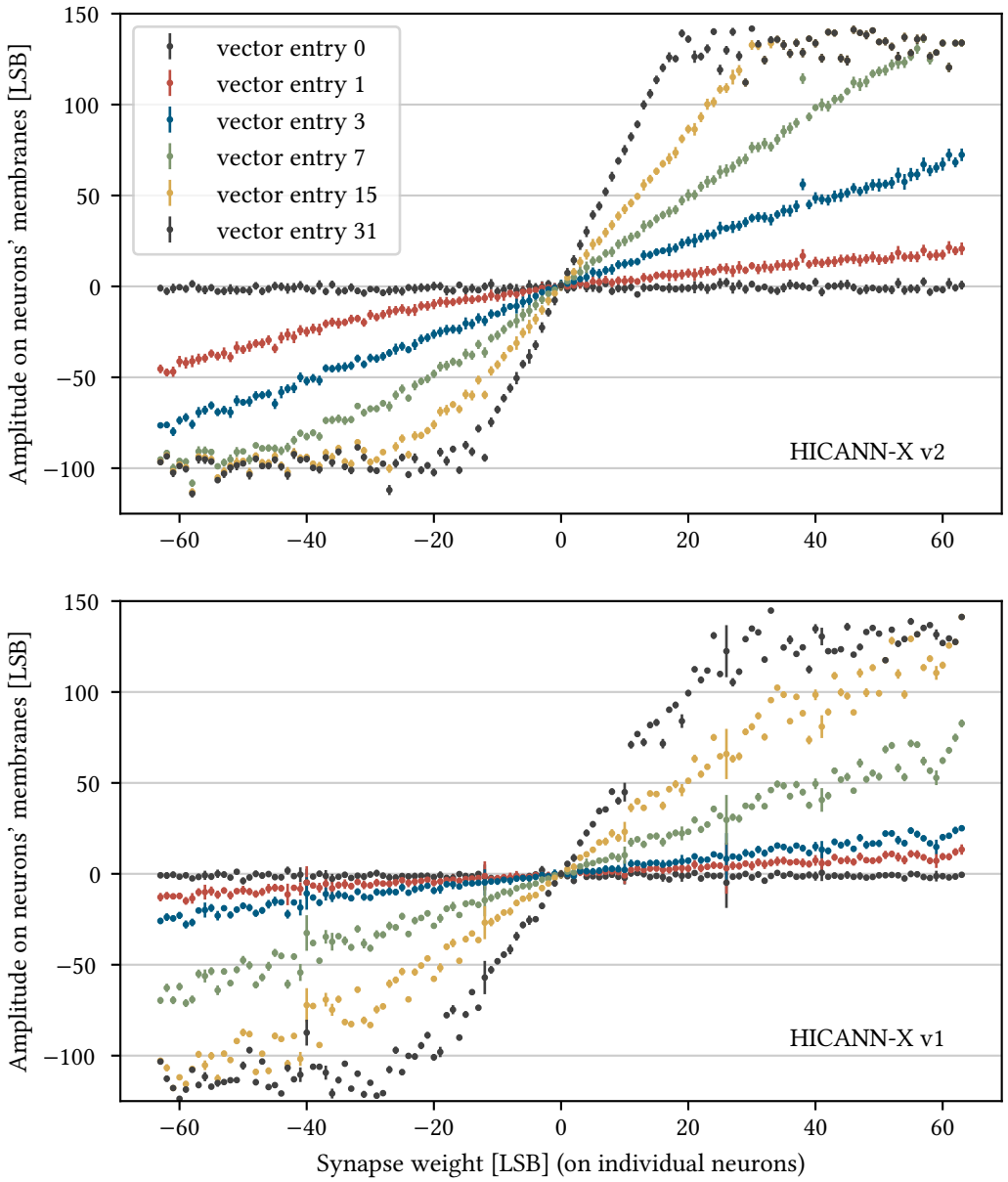


Figure 4.6: Characterization of linearity in both elements of the multiplication. The synapse weight increases per column from left to right, as indicated by the horizontal axis. **Top**: Measurement on HICANN-X **v2** (chip 23, setup 66). For low vector entries, amplitudes close to the synapse drivers are again higher, which is on the left side of this plot, since neurons 0 to 126 are plotted from left to right here. Linearity is generally good, but at lower amplitudes below −80 LSB, inhibitory inputs seem to get a bit weaker, approaching saturation. **Bottom**: Measurement on HICANN-X **v1** (chip 09, setup 69). Amplitudes are generally lower than on v2, and linearity is worse at higher input vector entries and matrix weights, as seen before with saturation of the synaptic input. There is no amplitude-dependent saturation visible, however. The inhibitory synaptic inputs approach the lower end of the digitization range more linearly.

right side of this plot. This effect is here amplified by the short pulse width and the high matrix weights in close proximity, even if further down the synapse row, the weights are zero, since the positive weight row is in use then. This also shows that the higher amplitudes close to the drivers are independent of the excitatory or inhibitory configuration of the synapse row. All these observations fit well to those using a constant weight matrix above.

For higher vector values, the amplitudes quickly start clipping at the dynamic range of the CADCs. The linearity in the middle of the plot, at lower matrix weights, looks good even at higher vector entries, however. For the negative weights, linearity looks a bit worse, with amplitudes below −80 LSB seemingly approaching saturation earlier. Some earlier clipping is expected since the neurons' baseline was calibrated at 120 LSB leaving a bit more room in the 8 bit range for positive results, but here, the minimum negative amplitudes seem to be scattered around −100 LSB. This is less dynamic range than expected. Also, the amplitudes seem to slowly approach clipping, instead of showing a linear characteristic until they reach a minimum. We might not see clipping in terms of the CADC input range at all. This suggests the inhibitory synaptic input behaves a bit differently in terms of saturation, or the neuron dynamics change at low membrane voltages, below some 0.3 V. Note that the inhibitory synaptic input OTA needs to operate close to ground potential there, which could lead to a decrease in gain, since with a small potential difference between the membrane and ground, it is more difficult to subtract current from the membrane. We will check this theory using a random weight matrix.

For HICANN-X v1, a similar plot is shown in the lower part of figure 4.6. The multiplication has a much lower gain here, the currently used settings yield roughly half the value of v2. Also, the fixed-pattern noise is much higher. This is in part due to leakage and the longer integration times, since wait times between two events are configured to 25 FPGA clock cycles here, instead of 8 on v2. The low vector entries seem to be matched quite well here, but remember this plot shows the mean amplitude emitted on all 128 synapse drivers, since all their amplitudes are integrated. After having investigated the drivers' pulse width generation in figure 3.5, we know these amplitudes are most likely the result of a few strong drivers and many other drivers emitting hardly any pulse. But the minimum inhibitory amplitudes actually look better than on v2: Indeed, the clipping happens later, with amplitudes scattered in the range of −110 LSB, some neurons reaching −120 LSB. Also, the approach to clipping does not look like gradual saturation, there is a sharper edge between linear operation and clipping. That said, the linearity looks generally worse, with slopes near zero matrix weights a bit larger than at higher amplitudes.

4.3.3 Random matrix

In order to investigate addition of positive and negative amplitudes, the final test during characterization of the MAC operations is a random matrix. This matrix is configured in the top right quadrant and is recorded while the linearity is characterized in the top left quadrant. We expect amplitudes to be generally lower for the random matrix, since with

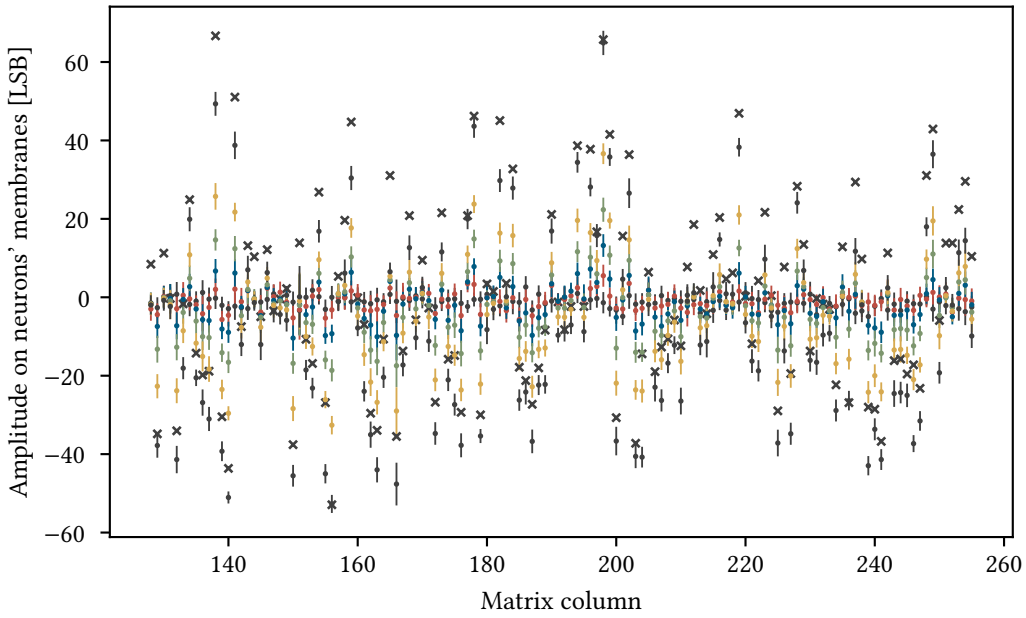


Figure 4.7: Amplitudes received at the neurons as result from multiplication between a random matrix and different constant vector entries, color-coded similar to figure 4.6. Since many signed random values in a column are added, amplitudes are generally closer to zero, and scaled with the vector entry, which works well. Executing the same random matrix multiplication on the host computer, and taking into account the gain measured at a constant weight matrix, lets us expect the amplitudes marked by the black crosses at a vector value of 31. Thus, the black measured dots should align with the black crosses. The systematically greater values of the black crosses mean that the inhibitory inputs are too strong, over-compensated during calibration at a lower membrane voltage. The measurement was taken on chip 23 (setup 66, HICANN-X v2).

a column of weights drawn uniformly from the available range, the expectancy value for the sum is close to zero. Some columns will have a slight positive bias, with greater values drawn, and some other columns will yield more negative values.

Figure 4.7 shows the amplitudes recorded on neurons using a random weight matrix. The plotted vector values are the same as for the linear matrix, with the maximum vector value of 31 indicated by a circular black marker, error bars again show the standard deviation across 30 runs. First, note that the order of the measured data points is only a scaling in the absolute value of the results, it does not change the sign. Low vector entries lead to values close to zero, but the green, yellow and black markers are either showing increasingly positive values or increasingly negative values. This is expected since the matrix weights stay the same and changing the vector values only changes a scaling factor, since the vector is constant. This shows that the synapse drivers work in the intended way:

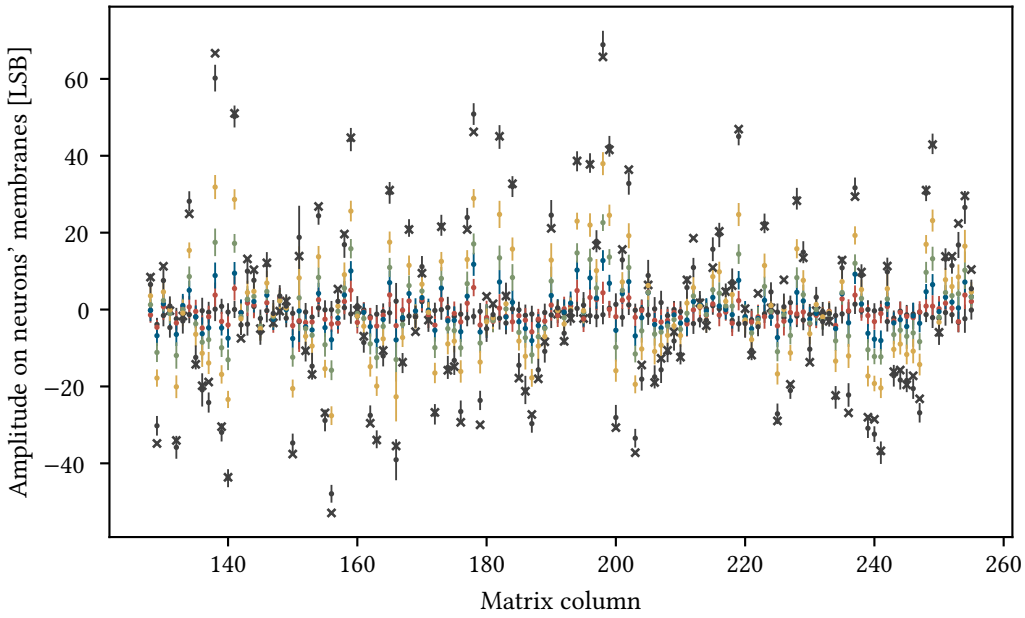


Figure 4.8: Amplitudes for the same random matrix multiplication as in figure 4.7, but the dynamic range of the CADCs was shifted to approximately 80 mV higher voltages. The non-linear range of the inhibitory synaptic input OTA at lower membrane voltages was avoided. The excitatory and inhibitory amplitudes are matched well here, but the characteristic is still not entirely linear, as shifting the potentials higher can increase the inhibitory synaptic input strength even further. However, these results should work well for practical usage. Compared to v1, the actual and expected results are much closer here, random variations are smaller. The measurement was taken on chip 23 (setup 66, HICANN-X v2).

If there were some synapse drivers which scaled the pulse widths significantly different from others, that would change the weighted sum of amplitudes.

Next, consider the black crosses in the plot, which mark the expected amplitude for this random matrix at maximum vector input, computed on CPU using the gain measured above. We quickly notice that most black crosses lay significantly higher than the black markers showing the actually measured value. A similar plot on HICANN-X v1 shows more randomness, but almost zero systematic differences across all neurons. There, the measurements lay 0.05 LSB higher than the expectations. These offsets are thus only a problem on v2, at least assuming the issue does not arise when using different chips, as I only tested one chip per version here. The differences can not be an error in measuring the gain, since that would only multiplicatively scale all amplitudes, and we see an almost constant offset here. In fact, the expected value is 8.9 LSB higher than the measured value in the mean of all 128 columns. This is a quite big effect, it would take a bit more than two synapses at full weight to compensate it, with this gain. Amplitudes expected to be below

zero are 8.0 LSB too high, those expected above zero are 9.9 LSB too high. This suggests excitatory and inhibitory amplitudes are in fact not acting on the membrane with the same strength and thus not added correctly here. The inhibitory synaptic input is too strong, decreasing the measured amplitudes.

Let's assume the inhibitory synaptic input OTAs indeed have trouble discharging the membrane at lower voltages. This would explain the early saturation of inhibitory amplitudes in the top part of figure 4.6 and also this effect: During calibration of the inhibitory synaptic input OTAs gain, the membrane voltage is lowered to some 0.3 V from its baseline at some 0.5 V. The early saturation, happening already there, would lead to a higher bias current for the inhibitory OTAs, in order to reach the same amplitudes as the excitatory inputs. But at higher voltages, near the 0.5 V baseline, the gain is now too high, and inhibitory inputs are stronger than excitatory ones. The fact that there even is a different shift for the lower and higher amplitudes in this plot suggests that even at higher voltages, the OTA is not entirely linear. It seems like for voltages above 0.5 V, it is even stronger, leading to a higher difference between expected and measured value, than for the voltages below that baseline.

In order to assess this theory, the experiment is repeated shifting the CADC's digitization range up by 45 LSB at the CapMem, which is approximately 80 mV. Since the neuron calibration utilizes the CADC, their baseline voltages follow the shift and the inhibitory synaptic inputs are calibrated at a higher membrane voltage. Otherwise, the exact same settings are used. Since the CADC is currently only calibrated with a CapMem cell, some offsets to the neuron baseline may be present between chips. However, the v2 chip already had a higher baseline voltage than the v1 one. In total, the neuron baselines are 110 mV higher than on the v1 chip used. With exactly this shift, the effect can be compensated, as shown in figure 4.8. The expected results are then only 0.2 LSB higher than the measured values in the mean of all neurons. Indeed, the results look very good. Plotting the linearity of multiplication again with this shifted baseline reveals that the inhibitory amplitudes indeed reach lower values now, with sharper clipping instead of gradual saturation.

While this suggests the issue can be easily mitigated using higher voltages, it is not completely solved. Shifting the neuron baseline voltage even higher, approximately 180 mV above its initial value, over-compensates the effect. The actual results are now higher than the expected values by 2.3 LSB. The non-linearity thus seems to be weaker, but it still extends to voltages of some 0.5 V, as already suggested by the different offsets for positive and negative amplitudes without the shift. On v1, the previous OTA design had no problems with lower membrane voltages. In fact, we did not hit the right voltage by chance, shifting the neurons' resting potential simply doesn't change the excitatory and inhibitory input strengths on v1: Decreasing the already lower baseline by 50 mV changed the difference between expected and actual results by only 0.003 LSB, which is entirely insignificant compared to differences of 8.7 LSB after a shift of 80 mV on v2.

Further investigating the issue, we look at the synaptic input OTA circuit. It uses a cascode current mirror [Bruun and Shah, 1995] to achieve independence of the current and the membrane potential. The gate voltage of these transistors, generated by the CapMem, needs to be set carefully. It was set to 340 LSB (roughly 0.6 V) for the measurements

above, where the systematic difference between expected and actual results was 8.9 LSB. Changing the cascode gate voltage to 300 LSB reduced the difference to 3.4 LSB at otherwise identical parameters, but a new calibration. While this suggests that indeed the effect arises from the OTA, the effect is weaker in simulation: Within a dynamic range of 0.3 V to 0.8 V, a change of 1.8 % in the output current is observed when setting the cascode voltage bias to 0.6 V [Sebastian Billaudelle, 2020, personal communication]. But we would need a difference of 3.8 % between excitatory and inhibitory inputs to explain the observations, and the range of membrane voltages during integration should not be as large. The simulation also confirms that a lower cascode gate voltage reduces the differences, but it must not be too low, as the transistors will lose saturation eventually. While the effect could thus be partly caused by the OTA, we have to keep in mind that shifting the neurons' membrane voltage affects many more components. These include the membrane capacitance or leak conductivity, and even the readout amplifier's characteristic, all of which could show changes with the membrane potential.

Concluding, the problem can be mitigated by choosing an appropriate neuron membrane voltage and cascode gate voltage on v2. However, as the effect never vanished, some training with hardware in the loop will be required to compensate it well, since receiving stronger inhibitory input initially will lead to slightly lower amplitudes of following inhibitory inputs. The effect depends on the membrane voltage and can therefore not be compensated for all possible inputs by changing the weight matrix, but it should be possible to compensate it well enough for a typical state, with distortions not too large in a sensible dynamic range.

5 MNIST benchmark

With hagen-mode working well on artificial test matrices, it is time to put it to a practical test. In order to benchmark the inference mode, two artificial neural networks targeted at classifying the MNIST dataset of handwritten digits [LeCun and Cortes, 1998] are used. This dataset contains 60 000 training and 10 000 test images shaped (28, 28), each pixel provides a single channel of 8 bit brightness information. After training on CPU, the software performance is measured at full precision and with weights discretized to the same range as on hardware. On hardware, for the first experiment, the same discretized weights are simply configured on a calibrated chip. Finally, training is continued with hardware in the loop, countering the distortions on chip.

5.1 The models

Since the reconfiguration of the weight matrix on hardware is expensive, convolutions are generally a good method to process larger amounts of data within the available dimensions of the synapse matrix. The MNIST images have become fairly simple to classify and do not require convolutions for high accuracies: Simard, Steinkraus, and Platt [2003] reach accuracies above 98 % with two dense layers, albeit even more using convolutional networks. The first and larger model will make use of a convolution layer, proving the operation can be implemented efficiently. Initially, the image is zero-padded by 1 pixel on each edge, shaping it (30, 30). The convolution uses 20 filters of shape (10, 10) and strides (5, 5), and a rectified linear unit (ReLU) activation function. The 500 results from convolution are processed in a dense layer with 128 neurons and a ReLU activation function. The final dense layer maps these 128 results to 10 neurons, representing the digit labels, and using a softmax activation function. We will call this model *the convolutional model*.

The second and smaller model contains only two dense layers. The first dense layer maps the flattened image as 784 inputs to 64 neurons and uses a ReLU activation function. The second layer maps the 64 results to 10 neurons, again using a softmax activation function. We will call this model *the dense model*. This model's weights fit entirely on one chip, requiring no reconfiguration at all. For the convolutional model, the second layer alone almost fills the chip, with its 500 inputs and 128 outputs.

Both models are trained in TensorFlow [Abadi et al., 2015] using 32 bit float weights. All layers are trained without biases. Weights are initialized with the Glorot uniform initializer [Glorot and Bengio, 2010]. During training, the Adam optimizer [Kingma and Ba, 2014] is used on a sparse categorical crossentropy loss function, with learning rate 1×10^{-3}. The convolutional model is trained for 20 epochs at batch size 32, the dense model is trained for 20 epochs at batch size 100.

5.2 Accuracy

	32 bit float	4 bit int	6 bit int
convolutional model:			
before re-training	98.29 %	97.61 %	98.07 %
after re-training	(98.531 ± 0.071) %	(98.312 ± 0.096) %	(98.403 ± 0.065) %
dense model:			
before re-training	97.43 %	96.99 %	97.36 %
after re-training	(97.42 ± 0.10) %	(96.99 ± 0.25) %	(97.472 ± 0.036) %

Table 5.1: Accuracy of the two presented MNIST models trained on CPU. After initial training of 20 epochs at full precision, results are evaluated on the held-out test set for different weight resolutions, the original 32 bit floats and 4 bit and 6 bit integers plus sign. The training is then continued for 10 epochs at the new precision. The mean and standard deviation of the accuracy on the test set after each of the 10 epochs is listed.

The models' performance is first investigated in software only, at full precision and at lower weight resolution. Since the high-level frontend for training with hardware in the loop, `hxtorch` [Spilger et al., 2020], as the name implies builds upon PyTorch [Paszke et al., 2019], I switched to PyTorch for all the further evaulations. Table 5.1 indicates the accuracy on the previously unseen test set of the models running on CPU, with and without re-training. The upper rows "without re-training" refers to the original weights from TensorFlow, which are discretized to lower resolutions by scaling them to the new range, without clipping, and using a symmetric floor function. The stated weight precisions mean integer weights plus a sign bit, so "6 bit" should represent an ideal hardware setup, using twin synapse rows for the signs. For the lower rows, the training was continued for 10 epochs at the new precision. The weights are stored as 32 bit floats internally, but casted to lower precision for each multiplication call. The gradients were not altered by the discretized multiplication function. The stated numbers are the mean and standard deviation of evaluations on the test set after each of the 10 epochs of continued training. At least to my eyes, there was no trend visible within these 10 epochs, the networks adjusted quickly within the first epoch of training.

Unsurprisingly, the larger convolutional model performs better than the smaller, dense model. For the convolutional model, we notice that even at full precision, the accuracy was still increased by continuing the training. This suggests the training previously ended in an unlucky state, performing worse than average. For the dense model, no significant change is visible in the accuracy. We should not yet have a problem with overfitting, since although the accuracy on the training dataset was higher than on the test dataset, the test accuracy did not decrease over the course of continued training. Before re-training, at full precision, the convolutional model yields an accuracy of 98.29 % on the test set and 99.45 % on the training data set. For the dense model, accuracy before re-training was 97.43 % on the test and 98.75 % on the training data. At lower weight precisions,

performance drops as expected, but not by much. After training with the new conditions, the convolutional model restores almost full performance. The dense model seems to be affected more, but only when reducing the precision to 4 bit plus sign. This could be due to the lower amount of weights - when drastically decreasing weight precision, larger models have more resources available for adjusting.

	HICANN-X v1		HICANN-X v2
	before re-training	trained on hardware	before re-training
convolutional model	92.13 %	98.01 %	93.79 %
dense mode	92.46 %	96.30 %	94.10 %

Table 5.2: Accuracy for both models on hardware, before and after training with hardware in the loop on HICANN-X v1, and yet only before training on hardware on v2. The listed numbers are results of only a single run. Standard deviations when repeating the experiment or switching to a different chip are estimated to be in the order of 2 %, but smaller in the trained state, even when using a new calibration. The measurements were taken on chip 09 (setup 69, HICANN-X v1) and chip 23 (setup 66, HICANN-X v2).

Using HICANN-X for the inference task results in accuracies given in table 5.2. Taking the models to calibrated hardware decreases performance due to statistical noise, remaining fixed-pattern deviations and distortions in the linearity of the operation. On v1, the convolutional model reached 92.13 % accuracy, the dense model 92.46 % before continuing training. Due to the trial-to-trial variations, which also change parameters of calibration by a few bits as shown in figure 3.7, the results from inference are not exactly reproducible. Switching to a different chip changes numbers even more, since the fixed-pattern noise changes and may fit the network better on one chip.

Unfortunately, using the hardware has been slower than on CPU, since on HICANN-X v1, a hardware bug required reconfiguration of the synapse matrix before every multiplication. This bug was fixed on v2, speeding up execution significantly. However, the execution of timing-critical code is difficult due to the communication to the FPGA, which only uses gigabit Ethernet. Buffering waits are required for precise timing and impose an additional runtime overhead. These waits ensure that the FPGA does not need to wait on instructions slowly coming in during real-time experiment control. Execution only reaches its target speed when experiment control is handled by the FPGA and the on-chip microprocessor (PPU), which are only passively involved currently. It is planned to support this directly from hxtorch, but this is not yet implemented.

The large runtime of up to several hours to classify the full MNIST test set is the reason why for the measurements on hardware, only a single number representing the accuracy of a single run is given. The standard deviation in accuracy when simply re-running the same experiment is in the order of 2 %. There are multiple reasons for these deviations, from the mentioned trial-to-trial variations during inference and calibration to networking delays and temperature. Optimizing the relevant parameters based on the mean result of multiple runs takes a lot of time and was only done for the dense network on v1. For the other

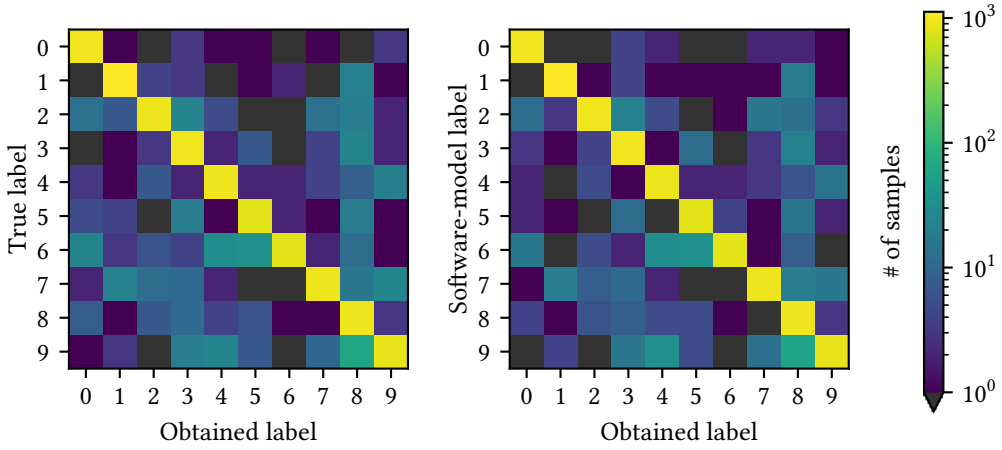


Figure 5.1: Confusion matrix for the dense model running MNIST on calibrated hardware, before re-training. The left matrix shows the usual confusion between obtained labels on hardware and the true results. The right matrix shows the results obtained on hardware with respect to the same model evaluated in software. We observe no severe problems with the classification. The experiment was run on chip 09 (setup 69, HICANN-X v1).

networks, the parameters are chosen suitably, but not necessarily optimal. Importantly, the parameters like vector resends and wait time between events were not optimized per layer, but only for the whole network. In principle, every layer has different typical input amplitudes and average weights, thus the constraints due to the synaptic input characteristics are different. This could be the reason why the convolutional model performs worse than the dense model on hardware before in-the-loop training, but it could also be coincidence with the fixed pattern deviations better fitting the dense model - the differences are not significant. Using identical hardware parameters and the original weights, all ten tested chips for HICANN-X v1 and both tested chips for v2 reached accuracies in the order of 90 %.

For the dense model, a confusion matrix on HICANN-X v1 before training on hardware is shown in figure 5.1. Note the logarithmic colorbar which enhances visibility of the wrong classification results. In the left matrix, the “standard” confusion matrix is shown, with respect to the true labels. The right matrix shows the confusion between the results on hardware and the results from the same model on CPU. This better characterizes the hardware since wrong predictions that would also have happened on CPU are not shown there. It is worth noting that while obviously more classification errors were introduced by using HICANN-X, in some cases a wrongly labelled image in the software model was correctly labelled on HICANN-X. This shows that the errors happen mostly if the model yields two labels at medium probability instead of a very clear prediction. Otherwise the confusion matrices look well, there are no systematic misclassifications visible.

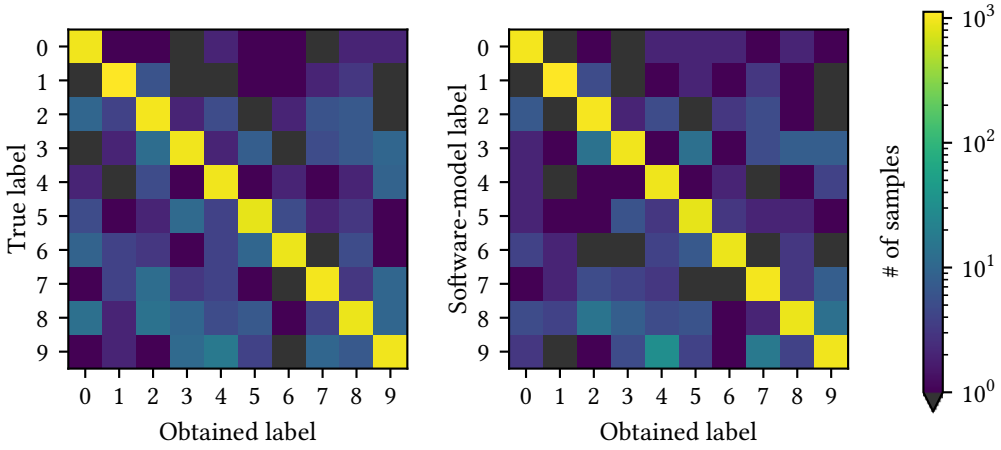


Figure 5.2: Confusion matrix for the dense model running MNIST after training with hardware in the loop for one epoch. The left matrix shows the usual confusion between obtained labels on hardware and the true results. The right matrix shows the results obtained on hardware with respect to the original model evaluated in software, before continuing training. As training on hardware counters fixed-pattern deviations, the results are better than without re-training, as expected. The experiment was run on chip 09 (setup 69, HICANN-X v1).

After training with hardware in the loop, the results have significantly improved. A new set of confusion matrices is shown in figure 5.2. Again, the left plot shows the confusion matrix with respect to the true labels, the right matrix shows the confusion with respect to the same model's performance on CPU. However, the weights are now not identically, they were trained for one epoch on hardware, at an unchanged learning rate of 1×10^{-3}, but an increased batch size of 200. The larger batch size provided more stability during training, since trial-to-trial variations leading to misclassification of a few images within the batch have less impact on the weight gradients computed for the whole batch. With lower batch sizes, the accuracy vanished in some cases, reaching chance-level and eventually starting to learn again from there. Larger batch sizes in the order of some 100 images are not a big problem in terms of runtime, since reconfiguring the weight matrix currently takes roughly 1000 times longer than processing a single vector. So unless a large convolutional network requires lots of vector multiplications with the same weights for a single image, the runtime is dominated by reconfiguration of the weight matrix for small batch sizes anyway.

Apart from the large runtime, training the dense model with hardware in the loop worked very well. The decrease in accuracy due to hardware distortions was countered within the first epoch of training, the accuracy stayed approximately constant for the remaining nine epochs. Before training, the accuracy for this run was $(92.71 \pm 0.15)\,\%$, with only a single evaluation run and the standard deviation taken from a different experiment,

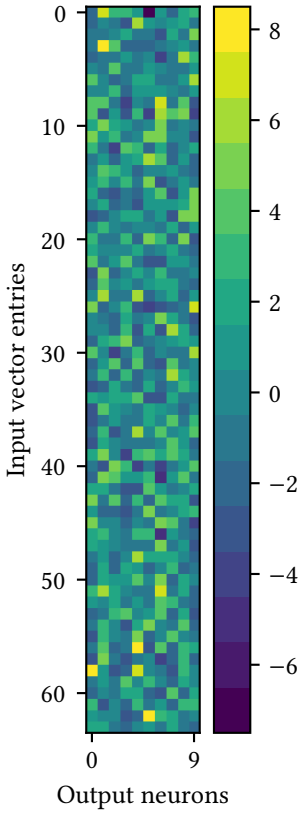


Figure 5.3: Difference of synaptic weights before and after one epoch of re-training on hardware. The weight matrix is the output layer of the dense model, receiving 64 inputs (in rows) and mapping them to 10 neurons (in columns). The colorbar shows the changes of the discretized 6 bit weights written to hardware. There is no obvious pattern observable, which suggests that at least the neuron calibration works well. Regarding the synapse drivers, which we know are not calibrated too well (see figure 3.5), we would expect a pattern of rows, but this could also be hidden in the previous layer. This experiment was done on chip 09 (setup 69, HICANN-X v1).

described in the next paragraph. By training with hardware in the loop, the accuracy was improved to (96.12 ± 0.16) %. This value is the mean and standard deviation across a single evaluation run after each of the 10 epochs trained.

Since the weight matrix now compensates fixed pattern noise on hardware, we take a look at the changes in the weights. In this run, using a different calibration but otherwise the same setup, a single epoch was trained, increasing the accuracy from (93.53 ± 0.15) % to (96.30 ± 0.09) %. The standard deviations are measured in 10 successive evaluation runs with identical parameters. Note the decreased standard deviation after training, which indicates the network was able to make close neuron activations more distinct. In the paragraph above, the training was continued for an epoch before each measurement, hence the larger standard deviation in accuracy after training.

Figure 5.3 shows the changes in discretized 6 bit integer weights on the chip for the output layer of the dense model after one epoch of training. It does not show a significant pattern. If the main problem were the neurons, we would expect a column-wise weight change, if the main problem were the synapse drivers, we would expect a row-wise pattern. However, the previous layers' columns could have absorbed a row-wise pattern, since the 64 output neurons from the previous layer are the vector entries for this layer, after applying the ReLU activation function. We do not recognize an obvious source for the fixed-pattern noise, but we might simply miss it in this plot. We will later re-visit

training with hardware in the loop while weakening the applied calibration, investigating if calibration can be replaced by training.

5.3 Optimizing hardware parameters

The analog multiply-accumulate operation has a lot of tunable parameters which affect its characteristics. Some of them are already used during calibration, like the target strength of the synaptic inputs. Some other are just set static and not even calibrated, like the synapse DAC bias current. Some important ones are set during the operation itself and configure how vector entries are sent to the chip, like the wait time between events. In order to gain better understanding of the parameters on the performance of a network, five of them are swept while classifying MNIST using the dense model on v1, without re-training on hardware. For each of the parameters, 16 settings are recorded, and 15 measurements per setting are taken on two chips. The many measurements with identical settings allow better distinguishing significant trends from random fluctuations. Using two chips reduces the effects of fixed pattern noise on accuracy, which may fit the network better on one chip. Both chips' results are presented independently since a sample size of two is far too little to calculate statistical moments. During acquisition of the data, the next run of measurements at identical settings was only started after all settings were measured once. This helps avoiding the absorption of dependencies on slowly changing external factors, like temperature, as a dependency on the settings. For each of the measurements, a new calibration run was executed, which we have observed to introduce more variability. For the accuracy measurement, only the first 2000 images of the MNIST test set were used instead of the 10 000 images available, due to runtime limitations - even as presented, the measurements took four weeks.

The five investigated parameters in this section are the target synaptic input bias current for calibration, the synapse DAC bias current, the number of vector sends during the integration phase, the wait time between two vector entries, and the membrane capacitance. Only one parameter was changed at a time, a default set was used for the others. These default parameters are a target synaptic input bias current of 100 LSB, a synapse DAC bias current of 1000 LSB, 6 vector sends, waiting 8 FPGA clock cycles between sending two vector entries, which is 64 ns, and maximum membrane capacitance of 63 LSB, which is approximately 2.2 pF. For the wait time between events, the experiment was repeated at a lower synapse DAC bias current of 600 LSB in order to assess benefits of mitigating synaptic input saturation this way. Finally, the cross-dependency between the number of vector sends and membrane capacitance was investigated by measuring a two-dimensional grid of both values, but only for 6 values per parameter, again due to the runtime. For all measurements, the parameters were set identically for both layers of the model. The second network layer, however, was concatenated two times within the synapse matrix, making the 64 inputs fill a vector of 128 entries. This means that the number of vector sends is effectively doubled, but using two different matrix rows on hardware has the additional advantage of averaging fixed-pattern noise from synapse

drivers and synapses a bit. This repetition was only used during this section sweeping the hardware parameters.

Synaptic input bias current. The target value for the synaptic input bias current directly affects the calibrated synaptic input OTA gains. Sweeping this value between 60 LSB to 200 LSB, no significant dependency between parameter and accuracy was measured. On both chips, the trial-to-trial fluctuations were larger than any trend. Since the synaptic input OTAs are the main source of noise on the membrane and, at a higher gain, yield more signal but also more noise, this is to be expected. Still, the assumption of the signal-to-noise ratio staying approximately constant when changing the OTAs' bias currents is therefore verified. I omit showing a plot of the measurement since it is not very informative and other parameters are more influential.

Synapse DAC bias current. The synapse DAC bias current scales the output currents of each synapse at a given weight. It thus controls the amount of charge reaching the neurons' synaptic input OTAs. This parameter's effects also do not justify a plot, but for the opposite reason: The accuracy decreases significantly when lowering the DAC bias current, since the synaptic input OTAs receive less signal. There seems to be no benefit from having less saturation, or waiting for 8 FPGA clock cycles between two events is already enough to avoid saturation. To investigate the latter, I also swept the wait time between events at this lower DAC bias current. Sadly, even when sending the events more quickly, the lower DAC bias current yielded absolutely no benefit. The accuracy is generally lower and even the drop in accuracy resulting from sending events too quickly seems not to be shifted to shorter times between the events. I expect that by increasing the number of vector sends while using weaker but faster inputs, the original result should be restored, but in that case, there is no benefit from doing so, since the integration time would stay the same.

Number of vector sends. The number of vector sends controls the signal received at the neurons' membranes, but, repeating the events, also affects integration time. We therefore expect a decrease in accuracy if the signal-to-noise ratio is too low using too few vector sends, but also an impact of using too many. Mainly, the neurons' membrane voltage will eventually exceed the dynamic range of the CADCs, making low and high positive inputs indistinguishable. This, however, only holds for integration times which are much shorter than the membrane time constant. Since every vector repetition requires time, this may no longer be a valid assumption. With the synaptic inputs configured rather weak and negatively signed inputs further decreasing the amplitudes, it is possible that the time-average of the current flowing onto the membrane is small enough to be compensated by leakage. Even continuously repeating the vector inputs, the membrane potential would reach a steady state. One could use this mode of operation to measure the synaptic currents, but this would require a calibration of the leak conductivity. The membrane time constant calibration will yield close values, but due to variations in the membrane capacitance, not identical values. While calibrating the leak conductivity is possible and even implemented already, measuring this average current on the neuron is not a good idea, most notably because of the required time to reach a steady state. Generally, we want to do the opposite and ideally turn off the leak OTA completely during integration.

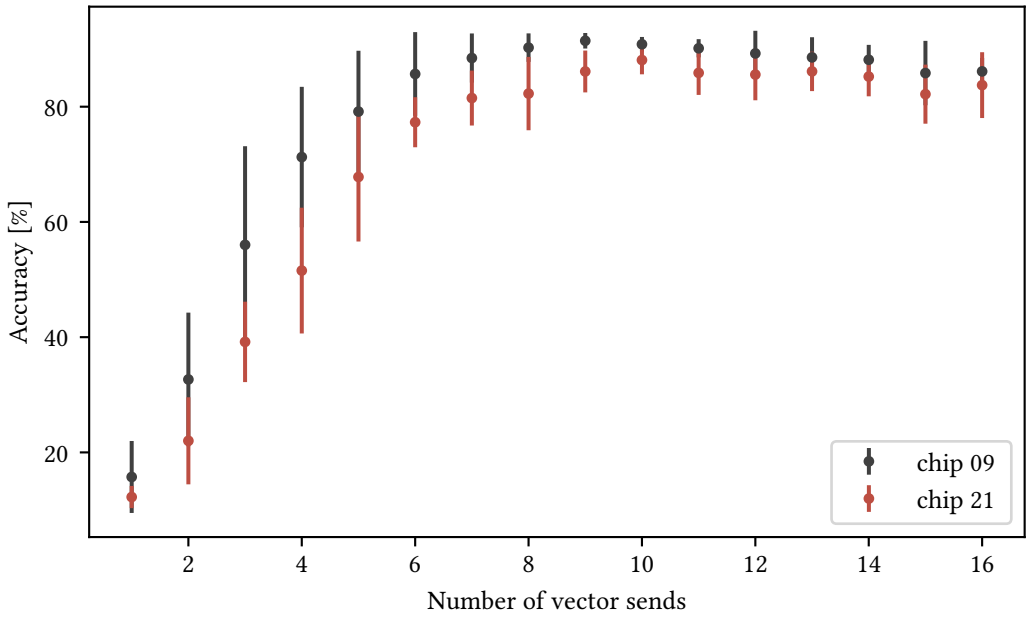


Figure 5.4: Accuracy of the dense model when sweeping the number of vector sends, showing mean and standard deviation of 15 runs on two chips (both HICANN-X v1). For too few resends, the signal on the membrane is too weak for accurate inference. After a maximum near 8 to 10 vector sends for this network, accuracy decreases slightly as amplitudes exceed the dynamic range for digitization and the required time for integration gets larger.

A plot of the accuracy when sweeping the number of vector sends is shown in figure 5.4. In the range of 8 to 10 vector sends, there is a maximum in accuracy for both chips. Using more repetitions, the accuracy decreases again, but not as severe. Already at 10 vector sends, integration times are quite high with respect to the membrane time constant: If the vectors contained the full 128 entries, integration would take 82 µs, larger than the typical membrane time constant of 60 µs. However, for this MNIST network, the first layer combines some rows of pixels into one vector, thus typically including a lot of zero values, most vectors have only some 40 to 60 non-zero entries. This effectively reduces the integration time by a factor of two to three, making it significantly shorter than the membrane time constant again. In the second layer, vectors typically contain some 35 to 45 non-zero entries within their size of 64, so even placing them two times in the vector of size 128, the membrane time constant is not exceeded even at 10 resends.

At the other end, towards using fewer vector sends, we observe that the accuracy decreases drastically. The signal-to-noise ratio on HICANN-X v1, at least for this network with the used settings, is insufficient there. While at 6 vector sends the mean accuracy has not yet decreased too much, the variability has significantly increased. The error bars, indicating the standard deviation of 15 experiments with different calibrations, are much larger than near the maximum. It seems like the network is much more sensitive to small

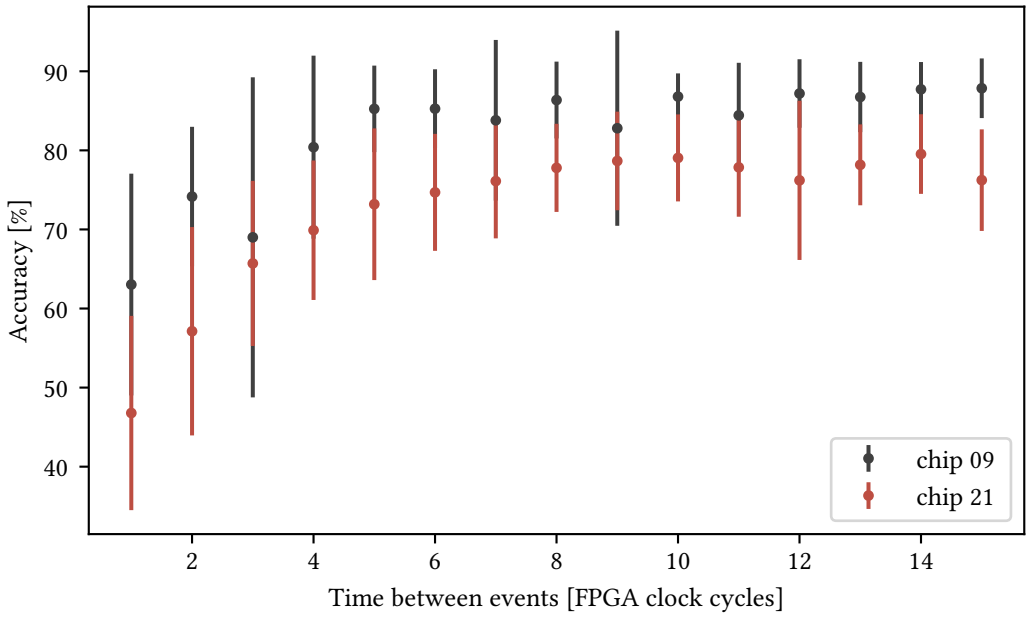


Figure 5.5: Accuracy of the dense model when sweeping the time between sending two successive vector entries to the chip, showing mean and standard deviation of 15 runs on two chips (both HICANN-X v1). When sending events too quickly, saturation of the synaptic inputs prevents accurate multiplication results. Above 6 to 8 FPGA clock cycles of wait, which is 48 ns to 64 ns, there are no significant changes in accuracy observable.

variations in the results from calibration there, where signal-to-noise ratio is just at the edge of a steeper decrease. As an alternative to using vector resends, the synaptic input time constant could be increased, which would, however, require longer waits between events in order to avoid saturation. While a benefit of doing so would be reduced input traffic, leakage would affect the first events much stronger than the last ones, distorting their amplitudes, as discussed before.

Wait time between events. The wait time between successive events mitigates saturation of the synaptic inputs, which generate less current than desired if events arrive too quickly. For a test matrix at constant weights, this saturation was investigated in figure 4.2. We expect that if the wait time is too small, the accuracy decreases as the multiplication operation is less linear. For higher wait times, the only issue is that integration time gets larger, so leakage is more relevant. Plotting the results in figure 5.5 reveals that indeed the accuracy decreases for small wait times, but seems to be constant for higher wait times above 8 FPGA clock cycles. This fits well to the observations with the constant weight matrix, where even for the lower amplitudes, no impact of leakage was visible at larger times between the events. However, the same test matrix on HICANN-X v2 shows this dependency, and generally, wait times should be as short as possible in order to minimize runtime. As already mentioned before, a similar sweep was done at a lower synapse DAC

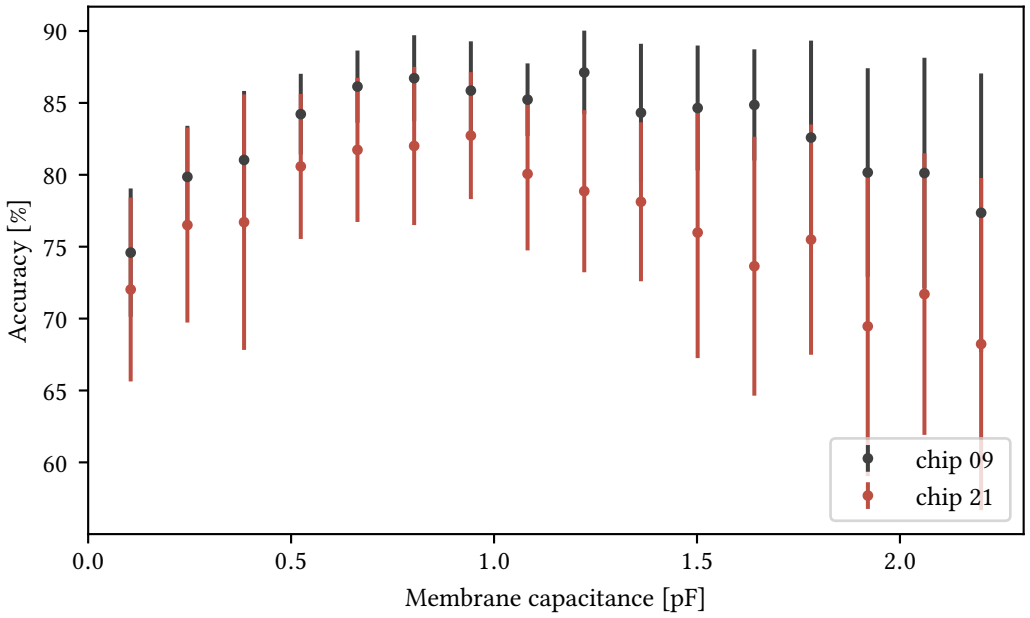


Figure 5.6: Accuracy of the dense model when sweeping the neurons' membrane capacitance, showing mean and standard deviation of 15 runs on two chips (both HICANN-X v1). A lower membrane capacitance makes the same amount of charge injected lead to a higher difference in membrane potential, but also decreases the time constant. There seems to be a maximum in accuracy near a capacitance of 0.8 pF. For lower capacitances, the accuracy is limited by leakage since the integration time is now longer than the membrane time constant. For higher capacitances, the membrane voltage rises less, and possible benefits regarding noise levels can not outweigh the loss in signal.

bias current, which never reached similar accuracy and did not even show less decrease in accuracy for lower wait times.

Membrane capacitance. The neuron membrane capacitance changes how currents emitted from the synaptic input OTAs affect the membrane potential. At a smaller capacitance, a small amount of charge will lead to a large difference in voltage. In practice, a small membrane capacitance increases the amplitudes of input events on the neuron's membrane, but also shortens the membrane time constant, if the leak conductivity is unchanged. Noise in principle gets amplified too, but there are dependencies on the frequency: Constant currents or very low frequency noise is not amplified due to the unchanged leak conductivity. Looking at the plot in figure 5.6, we observe that there is an advantage of using less than the maximum membrane capacitance. This implies that the signal-to-noise ratio is best there. For a lower capacitance, the membrane potential might clip at the dynamic range of the CADCs or the potential may decay too quickly due to leakage. At a larger capacitance, less amplitude is generated by the injected charge. In principle, the maximum at lower capacitance could be the same effect as with the number of vector sends, where the

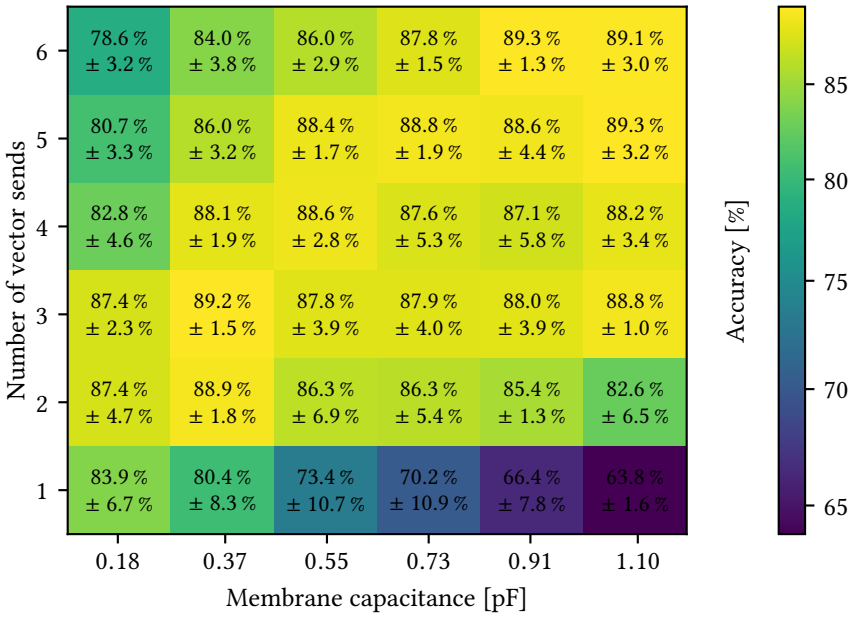


Figure 5.7: Accuracy of the dense model when sweeping both the neurons' membrane capacitance and the number of vector sends in the lower range of the respective parameters. The colors indicate the mean accuracy across 15 runs, the exact mean and standard deviation is printed for each point on the grid. Optimal accuracy can be maintained when reducing both the membrane capacitance and the number of vector sends, the diagonal of the plot shows good results. When using too few vector sends at a larger membrane capacitance, the signal is too weak and trial-to-trial noise distorts the results. When using too many vector sends at a low membrane capacitance, the amplitudes decay quickly due to the small membrane time constant or even reach the limits of the dynamic range of digitization. This measurement was conducted on chip 09 (setup 69, HICANN-X v1), but results from chip 21 (setup 73, HICANN-X v1) look similar.

chosen value of 6 is a little below the maximum accuracy. Using fewer vector sends, the optimal membrane capacitance could be even lower.

We conclude that the previously chosen default parameters are mostly ideal, naturally they were already chosen based on less systematic experiments. For the number of vector sends and the membrane capacitance, there is potential of optimization, since lower membrane capacitances seem to work well. We thus conduct a two-dimensional sweep of both parameters for 1 to 6 vector sends and a membrane capacitance of 5 LSB to 30 LSB in steps of 5 LSB, which is linearly mapped to physical values of 0.18 pF to 1.1 pF.

At each setting, again 15 measurements with newly executed calibrations are used to indicate mean and standard deviation of the accuracy. The other relevant parameters are chosen as for the sweeps above, i. e. the wait time between events is 8 FPGA clock cycles, the synapse DAC bias current is maximum and the synaptic input OTA bias current tar-

get is 100 LSB. Figure 5.7 shows the accuracy for each of the combinations of membrane capacitance and vector sends. Besides the color-coding of the mean accuracy, the mean and standard deviation of the 15 runs is given at each point. We observe that the optimal accuracy follows a diagonal line, from the lower left corner at 2 vector sends at a membrane capacitance of 0.37 pF to the upper right corner at 6 vector sends and a membrane capacitance of 0.91 pF. Within the optimum, the accuracy does not change significantly, so using 2 vector sends at a lower membrane capacitance yields the same results as using more vector sends at a higher capacitance. Towards the top left corner, the amplitudes on the membrane are too high, impacting accuracy, similarly to the decrease in accuracy when using too many vector sends in figure 5.4. For the bottom right corner, the signal-to-noise ratio is insufficient, and accuracy drops quickly, as observed before for using too few vector sends.

The diagonal optimum covers a certain range of membrane capacitance at a higher number of resends, which seems to get smaller at a lower number of resends. This makes sense since the membrane capacitance C_{mem} affects the amplitudes of incoming events by $\Delta V = \Delta Q / C_{\text{mem}}$, and this inverse proportionality means a steeper dependency at low values of C_{mem}. Going from the lowest to the second-lowest setting in the plot doubles the membrane capacitance, at least excluding parasitic capacitances, the following steps are smaller in a relative sense. It could be that the measured steps are too large to detect an optimum for 1 vector send, as the border between insufficient signal-to-noise ratio and too large amplitudes is more narrow. However, it also makes sense that vector resends generally benefit the results, since the differences in amplitude between the first and last events are not as strong if they are repeated multiple times.

The results plotted in the colormap were measured only on chip 09. The data obtained on chip 21 looks mostly similar, but this chip seems to require a bit more signal to achieve high accuracy, matching figure 5.4. Even at the second-lowest used membrane capacitance of 0.37 pF, 3 vector sends yield a 2 % higher accuracy than 2 sends in the mean of 15 runs. The diagonal thus appears slightly shifted, but generally the same effects are visible. We conclude that lowering the membrane capacitance can yield advantages in runtime since a lower number of vector sends is required to achieve similar accuracy.

With the network not trained on hardware, it reacts sensitively to changes in the hardware parameters and it may be necessary to optimize some of the considered settings for each network. We expect that after training with hardware in the loop, the impact of non-ideal parameters is smaller as the network can learn to cope with the resulting distortions. This would especially be the case if the hardware characteristics were considered directly, e.g., by modifying the loss function.

5.4 Long term stability

Alongside the nightly recording of calibration results (section 3.5), the accuracy of an MNIST network is measured on a nightly basis. The convolutional network is used for this task, simply because it existed already in December of 2019 when I set up the nightly

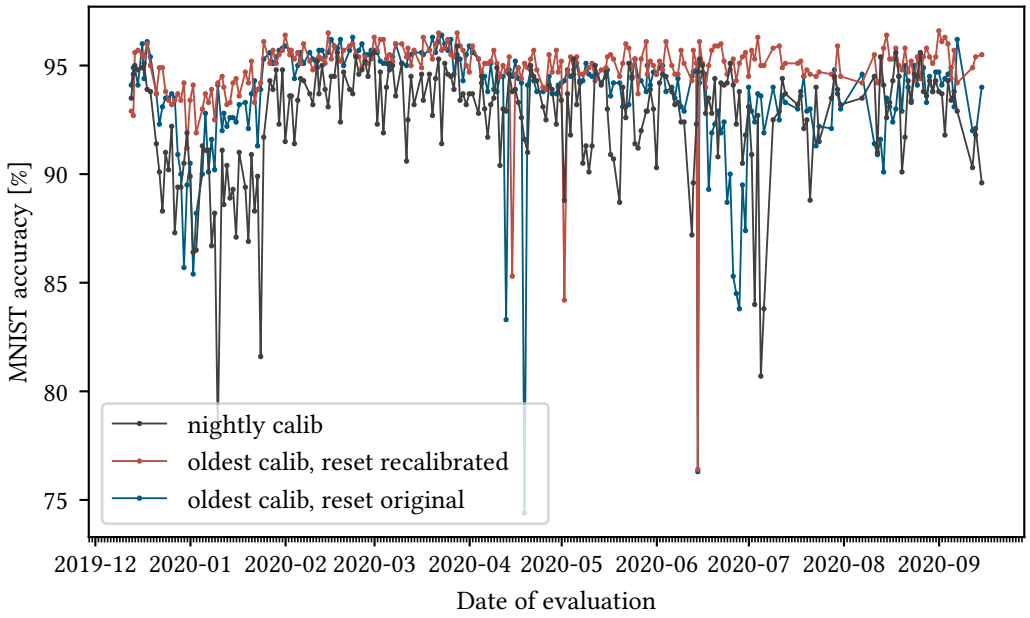


Figure 5.8: Accuracy of the convolutional network classifying the first 1000 images of the MNIST test set nightly since December 2019. Besides a newly acquired nightly calibration (black trace), one set of "old" parameters is continuously re-tested (blue trace), and again tested with re-calibrated neuron membrane reset potentials (red trace), which determine the potential before integration of inputs. We observe that the oldest calib with re-calibrated reset potentials continues to work well over eight months later. In its original state, the old calibration is more susceptible to external factors like temperature and the variability of the accuracy is higher. The nightly calibration shows the greatest variability. All other visible effects are interpreted in the text. The measurements were taken on chip 05 (setup 62, HICANN-X v1).

measurements. Only the first 1000 images of the MNIST test set are used, once again due to the otherwise large runtime. Different states of the calibration are evaluated: Firstly, the current nightly calibration is verified by running the network. Secondly, an old calibration from December 12th, 2019 is loaded in order to check if the setup still performs well using these settings. Further, the neurons' reset potential is recalibrated when using the old calibration, as the resting potential might shift when loading the parameters much later, when for example temperature has changed. Besides the old calibration originating from a single run, a "median" calibration is tested, where all parameters were calculated to be the mean of 10 calibration runs. As we have already observed some variance when re-running the calibration, this median calibration could perform better on average.

The accuracy of the nightly calibration and the old calibration with and without repeating the calibration of the reset potential is plotted over time in figure 5.8. The red trace, showing the highest and most stable accuracy, is using the old calibration with re-

calibrated reset potentials. The blue trace, which shows a bit lower accuracy and much more variability, is using the old calibration with the original reset potential, no recalibration. The black trace, at again a bit lower accuracy and high variability, is the result when using the nightly calibrations, running shortly before the classification. The median calibration is not included in the plot, its results were close to the ones from the old calibration, both in terms of mean accuracy and variability.

To put the obtained results into numbers, we treat accuracy measurements which are more than 7 % below the trace's mean as outliers and do not include them in the calculation of mean and standard deviation. Some reasoning is provided later, suspecting timing issues to be the cause of the sometimes low performance. This way, the nightly calibration yields an accuracy of $(92.9 \pm 2.0)\,\%$, the oldest calibration reaches $(95.0 \pm 0.8)\,\%$ if reset potentials are re-calibrated and $(94.0 \pm 1.6)\,\%$ if all parameters are unchanged. The old median calibration, where each parameter was calculated as the median of 10 calibration runs reaches $(94.9 \pm 1.0)\,\%$ with reset potentials re-calibrated.

Looking at the time evolution, we first notice that from mid of December 2019 to mid of January 2019, the mean accuracy of all calibrations is reduced with respect to the earlier and later results. In December 2019, I accidentally reduced the time between successive vector inputs from 8 to 4 FPGA clock cycles, and only noticed roughly a month later. Indeed, this would explain both the lower accuracy and the higher variability, we have seen both in figure 5.5. It seems that calibration results fitting the network less well impact accuracy more if wait times are shorter, where the multiply-accumulate operation is more distorted.

Secondly, the accuracy of all calibrations seems to drop again in late March 2020. I expect that the decreased mean accuracy is caused by an increase in temperature by roughly 2 K, which happened around this time, as the laboratory was equipped with five new host computers. The old calibration performs a bit worse from then on, as each transistors' characteristics change slightly. We would expect the nightly calibration to be affected less since it should simply find the new optimal parameters at the higher temperature. However, the variability has increased significantly, for all calibrations. Very prominently, there are some single measurements at accuracies below 85 %, but also above, the spread is now larger. I suspect that with the number of setups and users, which gradually increased since the winter, the network was more and more busy. We increased the number of HICANN-X setups from three in February 2020 to eleven in July 2020. Since my code did not include explicit waits in order for the FPGA to fill its buffers before the timing-critical integration, it could be that at busy times, sometimes a part of the input events is lost to leakage in a longer pause during integration. The fact that all calibrations (and mostly only one of them at a time) seem to yield low performance occasionally supports this assumption. It seems unlikely that some day only the old calibration with original reset potentials performs well, but another day, it requires recalibrated reset potentials to perform well.

Comparing the three calibrations, we notice that the black trace of nightly calibrations seems to perform worse on average than the oldest calib, which is surprising since the old calib was not special in any way. I suspect the noise optimization (section 3.3.4), which is

conducted in the end of the calibration on v1, to be the cause of this. I slightly increased the acceptable noise level there, and expected the benefit from the longer membrane time constants to outweigh the increase in statistical noise. It seems that this was wrong, and at least for this network with these settings, higher leak conductivities benefit the accuracy. It is possible that increasing the number of vector sends would have increased the mean result again. The bi-stability in some neurons' baseline potentials could also drastically affect the results of this leak optimization, since the transitions between the two stable potentials can happen seldom and might not be recognized when measuring the noise 60 times every 5 ms. This would lead to such neurons being set to a low leak conductivity, which may render them badly usable in phases where the constant currents emitted from the synaptic inputs charge the membrane.

The higher variability of the nightly calibration might also be caused by excluding neurons which failed to calibrate at least one parameter or were found to be too noisy. Not using these neurons was an attempt at generally increasing the accuracy, but it leads to a different placement of the matrix within the synapse array each time the set of excluded neurons changes. Since the fixed-pattern noise on the chip will fit the network better or worse depending on the exact placement of the matrix, I suspect part of the variability to arise from here. Despite all parameters of calibration were saved, I was unable to pinpoint exact correlation between the nightly accuracy and a specific parameter, which may only be one parameter of one neuron. However, most parameters after calibration were very stable, as shown for the synaptic input bias currents in figure 3.7. The less stable parameters are the leak bias currents, resulting from the noise optimization, and the synapse drivers' parameters - which I expect is an artifact of the low STP ramp currents leading to an insufficient range for calibration on HICANN-X v1.

My final guess in terms of reasons for the lower accuracy of the nightly calibration was that the chip was initially colder, slowly warmed up during the calibration, and processed the MNIST networks at a higher temperature. Especially for measuring the older calibration's performance, it was warmer, since I evaluated the nightly calibration first. To test this assumption, I implemented a second check of the nightly calibration's performance, which is executed in the end, after the usual inference runs are completed. The neurons' reset potential was calibrated again as temperature changes could affect it. However, the network's performance during this second evaluation run was very close to the initial evaluation of the nightly calibration's results. I therefore disproved this suspicion.

All the shown results were obtained without re-training the network on hardware. When doing so, it is advisable to continue using the old calibration anyway, since even small deviations in the remaining fixed-pattern noise will require a slightly different weight matrix for compensation. Therefore, the accuracy obtained with the old calibration is very important - and looks very good. The blue trace, using the exact settings obtained in December 2019, still performs a bit worse on average than the red trace, where the neurons' reset potential is recalibrated to be equal to the resting potential. With temperature, all properties change slightly, which means the resting potential of the neurons' membranes shifts, more than in spiking mode due to the low leak conductivity. The starting voltage before integration, which is the reset potential, would thus be significantly different from the

resting potential, and leakage during integration would affect the membrane differently than usual. This is the idea behind checking results with re-calibrated reset potentials in the first place, and it seems like this is a useful thing to do. Note however, that the goal is to disable leakage altogether, and when using a very precise timing of resets, integration and readout, this should be possible. The reset potential would then be the only relevant potential and differences from the resting potential would not be an issue. But the resting potential mainly shifts due to constant currents emitted from the synaptic input OTAs, where the optimal reference potential has slightly changed. These constant currents could still be an issue then and one would have to calibrate the synaptic input reference potentials instead. In any case, calibrating these potentials only takes seconds and we conclude that reusing an old calibration works well.

5.5 Replacing calibration by training

Most of the parameters covered by calibration are of technical nature, required for the vector-matrix multiplication process to work. Those include the reference potentials of the neurons' synaptic input OTAs, which are set such that the OTAs generate no current without input, or the membrane potentials, which have to be aligned with the dynamic range of the CADCs. However, other parameters can be compensated by the synapse weight matrix. The viability of replacing exact calibration by training has already been evaluated by Wunderlich et al. [2019]. We equalize the synaptic input bias current settings, as deviations in the strengths of the synaptic inputs across the neurons can be compensated by scaling the weights of each neuron's synapse column. Further, the leak bias currents are equalized, since different membrane time constants only affect the decay of amplitudes due to leakage, which is compensated by setting the first synapse rows' weights a bit higher than the last ones'. After the noise optimization at the end of calibration, the membrane time constants were different anyway, and should even become more equal again setting the leak bias currents equal. In order to investigate if the accuracy of a network can be retained even without calibration of these parameters, the calibration is gradually weakened while accuracy before and after training with hardware in the loop is recorded. In the final, "decalibrated" state, one measurement is taken with all neurons' parameters set to the median value of the calibrated distribution, and one additional measurement is taken after adding a random noise of ±5 LSB to the parameters, which prevents the CapMem from generating different currents than expected.

Regarding the strength of the synaptic inputs, the fact that also the synapse DAC bias current and the synaptic input time constants are not calibrated on HICANN-X v1 makes the distribution of synaptic input strengths much larger than if only the synaptic input OTAs' gain was getting decalibrated. As shown in figure 3.3, even on HICANN-X v2 with calibrated synaptic input time constants, the deviations across neurons are large, with more than a factor of 2 between the lowest and highest amplitudes. On v1, this uncalibrated distribution is even larger, reaching roughly a factor of 4 between smallest and largest amplitudes. This means that for replacing calibration, the required dynamic range

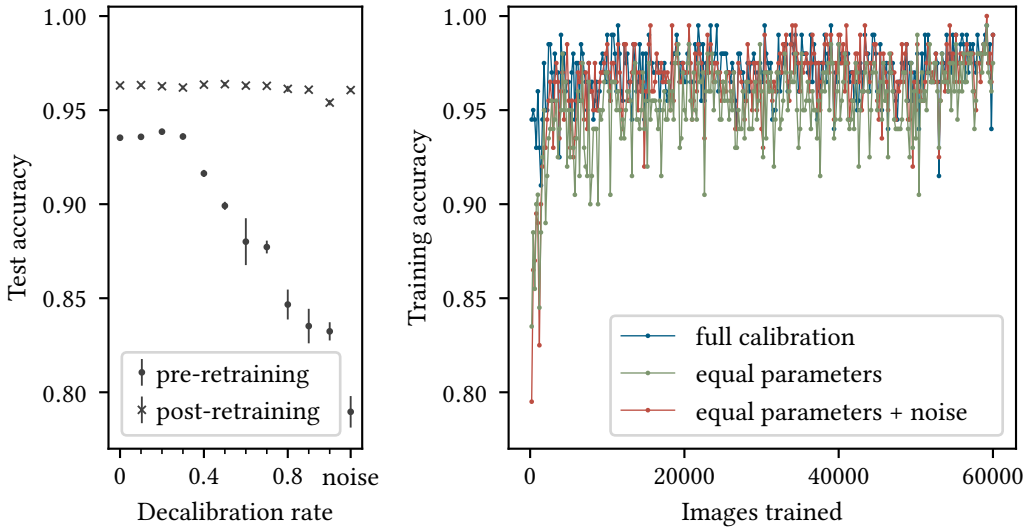


Figure 5.9: Accuracy of the dense model classifying MNIST at weaker calibration. The left part shows the accuracy on the test set before and after one epoch of training with hardware in the loop, with mean and standard deviation of ten identical runs per data point. The neurons' synaptic input and leak bias currents are slowly shifted from calibrated state (decalibration rate 0.0) to all the median values (decalibration rate 1.0). Since the CapMem has trouble handling many equal parameters, the rightmost data point shows the median values plus ±5 LSB of noise. The right part of the figure shows the accuracy per batch of 200 images during one epoch of training on hardware. The network adjusts quickly in either state of calibration, but can not mitigate the problems when setting all parameters equal, as evident by the lower accuracy of the green trace. The measurements were taken on chip 09 (setup 69, HICANN-X v1).

of weights is this factor of 4, or 2 bit of the synaptic weights. The weight precision available to the network is thus essentially reduced from 6 bit to only 4 bit. We therefore expect a slight performance drop as seen for the model evaluated on CPU in table 5.1. The dense network, which will be used for the experiment, was able to cope well with the reduced weight precision and dropped in accuracy by 0.37 %.

The accuracy when gradually weakening calibration is plotted in the left part of figure 5.9, before and after training with hardware in the loop. The markers show mean and standard deviation of 10 runs at identical parameters. We can see that the accuracy before re-training on hardware (marked by black dots) decreases as calibrated parameters approach their mean value, but the drop is less severe than I had expected, considering the spread of uncalibrated amplitudes (cf. figure 3.3 for a histogram measured on HICANN-X v2). The network seems to be quite robust against some of the fixed pattern noise. After one epoch of training, the accuracy (marked by black crosses) almost reaches the same value as for the calibrated chip, regardless of the state of calibration. Only when setting all

parameters exactly equal at a decalibration rate of 1.0, the CapMem issue (section 3.1) leads to worse results. Since the accuracy after adding some noise to the parameters is higher again, the issue not with the fixed-pattern noise itself. The decrease in accuracy from full calibration to full decalibration with noise is (0.24 ± 0.14) %, which is a smaller drop as for the quantisation to 4 bit weights in software, but the difference is not significant. Note that all used neurons were placed within one quadrant of the chip, so deviations in the synapse DAC bias current did not affect the results here, leading to a potentially smaller distribution of neurons' amplitudes than estimated before.

The right part of figure 5.9 shows the accuracy during training at different states of calibration. For each trained batch of 200 images, the accuracy on this part of the training data is plotted. An evaluation on the test set was once more not possible due to runtime limitations, with the measurement running for weeks as presented. The blue and red traces show the course of training at full calibration and full decalibration, the latter meaning median parameters plus ±5 LSB of noise. We observe little difference between the curves, the blue trace starts at a higher accuracy than the red one, but both quickly reach a higher accuracy. The green trace is measured with all neurons' parameters set to the median of the calibrated parameters, without noise. The CapMem introduces further distortions here, which seem to prevent the accuracy from reaching higher values. Although starting higher than the red trace, the green trace stays significantly below the other two. However, this is only a technical issue and we conclude that training with hardware in the loop can indeed replace calibration, at the expense of weight precision. In practical usage, calibration has the advantage of working for all networks, while training with hardware in the loop would need to be extended for each network in order to compensate the larger deviations.

The last experiment regarding the interaction of calibration and in-the-loop training is using the weights obtained after training on hardware with a different calibration. The new calibratios use the same settings, but will yield slightly different results. For the old calibration, no parameters were adjusted when re-evaluating the network accuracy. I took 20 measurements of accuracy in each case in an alternating pattern, so that varying external conditions should not introduce a bias to the results. With the original calibration, the accuracy was (96.20 ± 0.16) % within 20 runs of the dense model. With different calibration parameters, the accuracy was reduced only slightly, reaching (95.83 ± 0.17) % after 20 runs, each with a separate new calibration. The difference is significant, but the accuracy using new calibrations is still much higher than using the original weights from CPU. It seems the changes to the resulting fixed-pattern noise introduced by using a different calibration are small, or at least the network is more robust against these perturbations with the updated weights. Indeed, also the variability between the 20 different calibrations is an order of magnitude smaller than during the nightly evaluations. However, these were done using a different network on a different setup with possibly less ideally chosen hardware parameters, which we have observed to increase variability. Also, the 40 measurements presented here were all done within one day, which may decrease variability as opposed to the nightly measurements covering more than eight months. Still, we conclude that training with hardware in the loop yields consistent benefits, even when using a new cal-

ibration, which should also allow reliably using the same chip with the same weights in different external conditions.

5.6 Energy consumption

The main motivation for using analog hardware for inference tasks is its energy efficiency. The currently used setup consists of a host computer, an FPGA and the HICANN-X chip itself. The power consumption of host computer and FPGA is large compared to the chip, with only the FPGA using some 10 W. There are more energy-efficient solutions being designed, which will require only a few watts for the whole setup. The chip itself was measured to use approximately 0.3 W [Cramer et al., 2020] in a similar setup to what we use, with the CADCs reading out the neurons' membrane potentials. However, the number of active links between chip and FPGA affects power consumption. With the low event rate used on HICANN-X v1 due to the longer synaptic input time constants, two active links suffice, which is where the 0.3 W were measured. Increasing the event rate to its maximum, where two events are sent per FPGA clock cycle, all eight links are required, which results in a power consumption of roughly 0.36 W [Sebastian Billaudelle, 2020, personal communication]. The chip's power consumption is mostly constant, i. e. there is little difference between running inference and idling with similar configuration. We therefore calculate the energy consumption based on these power measurements and an estimation of the required runtime.

Currently, experiment control is entirely based on the host computer. This leads to a large increase in runtime, since commands like neuron resets and CADC reads need to be handled one after another for each neuron instead of using the PPU's vector unit for parallelism. Table 5.3 provides an overview of runtime with efficient implementation (hardware capability) and measurements of the current software state. A single MAC operation should take no longer than 5 µs when implemented properly. Currently, we need approximately 2 ms for the operation as available from the `hxtorch` frontend and measured on HICANN-X v2. This is much more than the sum of its three components above. Each individual step was timed on the FPGA, where the commands were stored in advance. For the MAC operation, however, all overhead including compiling the instructions, sending them to the FPGA via Ethernet, running the program and decoding the results are included. Although a batch of 1000 vectors was used, the current measurement is still slower than anticipated. With 91 %, the largest part of the runtime was spent during execution on hardware, including networking - which is what we expect to contribute most to the measured runtime. Only for the actual multiplication and accumulation phase, the measured runtime is equal to the optimum implementation, since the FPGA acts as a real-time experiment controller here. We verified the analog circuitry operates as expected and is capable of using the claimed timings in the left column using an oscilloscope. Further, configuring the synapse weight matrix from the FPGA via Omnibus was benchmarked in simulation and on hardware (chip 25, setup 60, HICANN-X v2) to take 1.4 ms of time [Vitali Karasenko, 2020, personal communication].

	hardware capability	current software
Neuron reset	0.5 µs	10.6 µs
Multiplication and integration	3 µs	3 µs
CADC read	1.5 µs	10.9 µs
Full-chip MAC	5 µs	2 ms

Table 5.3: Break-down of runtime for a single MAC operation, assuming full chip utilization. For each step, we indicate two durations: The time on the left is what the hardware can achieve with efficient experiment control handled by the PPU. The feasibility was checked using an oscilloscope. The values on the right represent the currently achieved performance using host-based software for control flow, which introduces additional overhead. They were acquired on chip 22 (setup 63, HICANN-X v2).

For the following estimation of energy efficiency, we will use the values that should be reachable on hardware. A single full-chip MAC operation is expected to take roughly 5 µs. It starts with neuron resets, which require 1 µs at full membrane capacitance. As we have seen in figure 5.7, reducing the membrane capacitance is possible without loss of accuracy, which enables faster reset times due to the smaller amount of charge which needs to be transferred during reset. Hence, we use 0.5 µs for the neuron resets in this estimation. The actual multiplication and integration of charge takes less than 3 µs in a typical case on HICANN-X v2. It was shown in the bottom part of figure 4.2 that this time suffices for reaching sensible amplitudes at the neurons without saturation of the synaptic inputs. The number is further supported by early experiments training MNIST on HICANN-X v2, where high accuracies were reached despite these low integration times. After integration, we wait for 0.5 µs for synaptic currents to finish charging the membrane. The CADC readout can be triggered already earlier, since digitization also requires a bit of preparation, such as resetting the ramp voltage, which takes 0.4 µs. The CADC ramp then runs for 1 µs and digitizes the voltages. Here, some time can be saved by using a steeper ramp, sacrificing digital precision of the result.

Within this time of 5 µs, the chip uses 1.8 µJ of energy at a power consumption of 0.36 W. Using all the available resources and counting each synapse as one multiplication, the chip is therefore capable of processing 26×10^9 multiplications and accumulations per second in unsigned mode, returning the accumulated results. The energy consumption per single multiplication and accumulation is thus 14 pJ. Counting both multiplication and accumulation as two operations, the chip's power efficiency is approximately 146 GOPS per watt. In signed mode, the throughput is only half, since we can use only one of two synapses. However, the label bit allows more efficient placement of network layers then, as will be shown for the two MNIST networks. Generally, all the numbers do not include the initial configuration of matrix weights, which requires 1.4 ms. When processing a large batch of vectors without changing the weights, this time gets negligible. However, a bug on HICANN-X v1 made it necessary to repeat part of this configuration for each vector,

thus all the numbers were increased by a factor of almost 1000. This bug has been fixed on v2.

The convolutional network classifying MNIST has three layers. The first convolution needs to process 25 vectors with all the same weights. The kernel is only shaped (100, 20), thus it can be placed multiple times on the chip, two times per synapse array, utilizing the synapse label bit. The training needs to consider the different placement of the four kernels which would normally be identical, and update their weights independently to compensate fixed pattern noise. It is advisable to learn with all vectors on all instances of the convolution, since otherwise the network would change architecturally, with each instance independently processing a part of the data. Doing all that, 6.25 full-chip MAC operations are required to process all convolutions for one image. The second layer, a dense layer shaped (500, 128) can be configured using all four quadrants of the chip as the four independent matrix parts mentioned before. The final dense layer shaped (128, 10) can be processed during one of the convolution operations, since it fits easily besides the other weights in one synapse array. This naturally requires batched execution of the network since the final layer's input data has to be ready then. But assuming all the mentioned parallelization is done and batch sizes are large enough for weight reconfiguration to be negligible, classification of one image requires eight MAC operations, which equates to 40 µs of runtime or 14 µJ of energy. For comparison, the runtime on a quad-core CPU was 37 µs per image, but assuming the CPU used its thermal design power of 84 W [Intel i7-4771, 2013], the energy consumption is three orders of magnitude higher at 3.1 mJ per image.

Experiments on HICANN-X v2 have shown that the previously claimed low integration times can be achieved: For the convolution, we use two vector sends and two FPGA clock cycles of time between events. The first dense layer also uses two vector sends and two FPGA clock cycles of time between the entries. The final dense layer uses four vector sends but only one FPGA clock cycle between the events, so not waiting at all. For all these layers, a full-size vector of 128 entries needs 4 µs to be processed. However, skipping the vector entries of zero saves a significant amount of time. For the convolution, a vector is typically filled with not more than 75 non-zero entries, which yields integration times of up to 2.4 µs. A similar argument holds for the two dense layers, where vectors are mostly filled with 40 to 70 entries. With this setup, accuracies above 97.5 % were reached on chip 22 (setup 63, HICANN-X v2). While therefore the numbers stated above are reasonable, the parallel execution of convolutions is not yet supported by the software and was not used when measuring the mentioned accuracies.

The dense network, with its two layers, fits completely on the chip. The first layer, shaped (784, 64) can be placed in seven parts, each shaped (128, 64). This means that 64 neurons are still available and 10 of them can be used for the final layer, which is (64, 10). In this configuration, each of the two synapse matrices is split into four parts, each requiring independent vector inputs. Since each of the four parts requires full-size vector inputs, we can not further parallelize them using non-overlapping pacement, and can only process two parts at a time using the synapse label bit. The synapses which are not targeted will still receive inputs and do their multiplications, but the results will be discarded.

Again assuming batched execution, processing one image only requires two full-chip MAC operations: The first one processing four of the seven parts of the first layer, the second MAC processing the remaining three parts of the first layer and the second layer for the image before. With these optimizations, classification requires 10 µs of runtime or 3.6 µJ of energy. For this network, a quad-core CPU is even slower than the chip at 20 µs per image, and, again assuming it uses the thermal design power, is much less energy efficient at 1.7 mJ per image.

For the experiment on HICANN-X v2, the first layer used two vector sends with events every two FPGA clock cycles. The second layer used four vector sends without wait, so with one event every FPGA clock cycle. As above, this means a vector of 128 entries would need 4 µs to be transmitted, but again, they contain a significant amount of zeros. For the second layer, the input size is only 64, thus the integration time is already cut in half. The vectors typically contain up to 60 non-zero entries in the first layer and 45 non-zero entries in the second layer. Hence, the integration time typically does not exceed 2 µs, well covered by the above estimates. With these settings, accuracies above 95.5 % have been reached on chip 23 (setup 66, HICANN-X v2). There is still room for optimization as these are early results, but we can conclude that a good energy efficiency can be achieved. Speeding the MAC operations up further will require hardware changes, as currently, even the neuron resets and digitization take some 2 µs. For example, introducing a sample-and-hold mechanism for digitization would enable pipelining since the neurons could already be reset and start processing the next vector while the CADC ramps are running.

Ultimately, we have to say that while the HICANN-X chip is much more energy efficient than using a CPU, this comparison is not entirely fair. Firstly, the complete setup around the chip will multiply power consumption, even if we consider an efficient solution that is currently being developed. Secondly, the CPU used was released seven years ago, and efficiency has improved since. Most importantly, a CPU is not the state of the art when it comes to efficient inference. GPUs yield large floating point processing power and more specialized digital hardware like Google's TPU promises a good energy efficiency. An Edge TPU, optimized for low-power devices, reaches a power efficiency of 2 TOPS per watt [Coral Edge TPU, 2020]. This is more than an order of magnitude better than the estimate of 0.15 TOPS per watt for HICANN-X v2. In order to be competitive, we will need optimizations in hardware to achieve a higher processing rate of vectors. However, using a hybrid operating mode of spiking networks and matrix multiplication could still yield advantages. The possibility of, e. g., using convolutions to preprocess large amounts of non-spiking input data and have a spiking layer obtain the final result is compelling.

6 ECG analysis

The current higher-level goal for inference on HICANN-X is the classification of ECG traces. The regularity of heartbeats and the existence of a P wave can be used to distinguish normal (sinus) rhythm and atrial fibrillation, as described in section 1. The available ECG data consists of two channels which show the voltage differences between three points on the body for 120 s. The dataset contains 16 000 traces, 8000 of each label. They are sampled at a rate of 512 Hz and saved in a format which requires a bit of preprocessing in order to get suitable 5 bit vector entries. Most importantly, the chip does not support signed vector inputs, at least only if four synapses were combined to allow both signed weights and signed inputs. We work around this by applying a delta-encoding as part of the preprocessing, which will be handled by the FPGA. This results in a trace which is centered around zero, the highest positive input vector activations appear at the rising edge of a heartbeat. Negative values are now simply cut off.

The currently preferred solution for ECG classification is using an artificial neural network with all the matrix multiplications handled in hagen mode. The network design follows a similar approach to the Human Activity Recognition task presented in Spilger et al. [2020]. The preprocessed data is sent to a convolutional layer, where the one-dimensional convolution kernel is placed multiple times within one synapse matrix, strided in the same way the input would be strided. This way, a larger input vector can be used to execute multiple convolutions at the same time, introducing the same problems with compensating fixed pattern noise as mentioned before, since intrinsically identical weights are now to be updated differently. The outputs of the convolutional layer are processed by two dense layers to yield the classification result. All shapes are chosen such that the whole network fits onto the chip. Designing the ideal network for the task is a whole new Master's thesis, but I can say that the current implementation reaches an accuracy close to the competition's target when run on HICANN-X v1 [Arne Emmel, 2020, personal communication]. This target accuracy is correctly labelling 90 % of the traces with atrial fibrillation and 80 % of the traces with sinus rhythm.

Once this accuracy requirement is fulfilled, the competition awards the most energy-efficient solution. In terms of runtime, classifying the ECG trace using hagen mode has the advantage of not relying on analog emulation in the chip. A vector is simply processed as fast as possible, which should again be in the order of 5 µs. With some downsampling and using only part of the data, we expect that the number of required vectors is not too high, we may only need two vectors per trace. This would mean processing times in the order of 10 µs per trace.

In the following, I will present two fundamentally different approaches at classifying the ECG traces, using the analog circuitry in spiking mode. There, we are limited by the analog time constants and can not process the traces arbitrarily fast. For both cases, I

chose a speed-up of 1×10^5 compared to real-time of the ECG traces. While some further speedup of a factor 2 to 3 may be possible, finding another order of magnitude is most likely not. Processing all 120 s of trace would thus require approximately 1.2 ms of runtime. The chip's power consumption is dominated by idle power, therefore the solutions would not be more energy-efficient than using hagen mode. However, we can potentially parallelize the processing of many traces, more than ten, making up for the larger runtime per trace. Also, the chip's energy consumption may decrease slightly if we do not enable the neurons' readout amplifiers, which are only necessary for the CADC readout in hagen mode. A drawback is the increased amount of preprocessing required, since the ECG signal is sent to the chip as a spiketrain, with a spike at every heart beat.

I will investigate classification of the ECG signals using the synapses' correlation sensors and using short term synaptic plasticity. All the presented work in this section is more a proof of concept than an optimized and well-characterized solution for the ECG task, since we focus on the approach using convolutional networks.

A simple benchmark classification method is a Poincaré plot, also called Lorenz plot or successive RR plot, where the time between two heartbeats is analyzed for its randomness [Anan et al., 1990; Sarkar, Ritscher, and Mehra, 2008]. There, one such time interval is plotted on one axis, and the following interval, the time until the next heartbeat, on the other axis. In case the points lay close to the diagonal, the heart beats regularly and the person is classified healthy. In case the data points are scattered more, the person is labelled to have atrial fibrillation. This method reaches an accuracy of some 91 % on the given dataset [Arne Emmel, 2020, personal communication], which marks a reference for the two hybrid solutions on chip.

6.1 Correlation sensors

With the ECG signal converted to a spike train, the synapses' correlation sensors can be used in order to find out whether the incoming spikes are regular. For that, multiple neurons are configured to be spiking regularly, with the leak potential set higher than the spike threshold. Using different spike frequencies at the neurons, different heart rates from $50\,\mathrm{min}^{-1}$ to $120\,\mathrm{min}^{-1}$ were tried to be correlated to the incoming spiketrain. If the ECG trace contains a regular heartbeat, a high correlation between the input spikes and one neuron's reference spikes should be measured. Over a certain period of time, the differences in spike time between a neuron and the input spikes should yield a random value for all neurons but one, which yields a high causal or acausal correlation. Since the phase is not known, we do not know whether there will be more causal or more acausal correlation, but the times should not be sampled randomly at least.

During the experiment, the correlation sensors' accumulation capacitors are initially reset to a high voltage and a baseline read is taken using the CADCs. Then, a section of 5 s of the ECG trace is sent to the chip as spikes. Each synapse within the row individually measures the exponentially weighted differences in spike timings between the input signal and the neurons' reference spikes, as shown in figure 2.5. After the batch of spikes is sent,

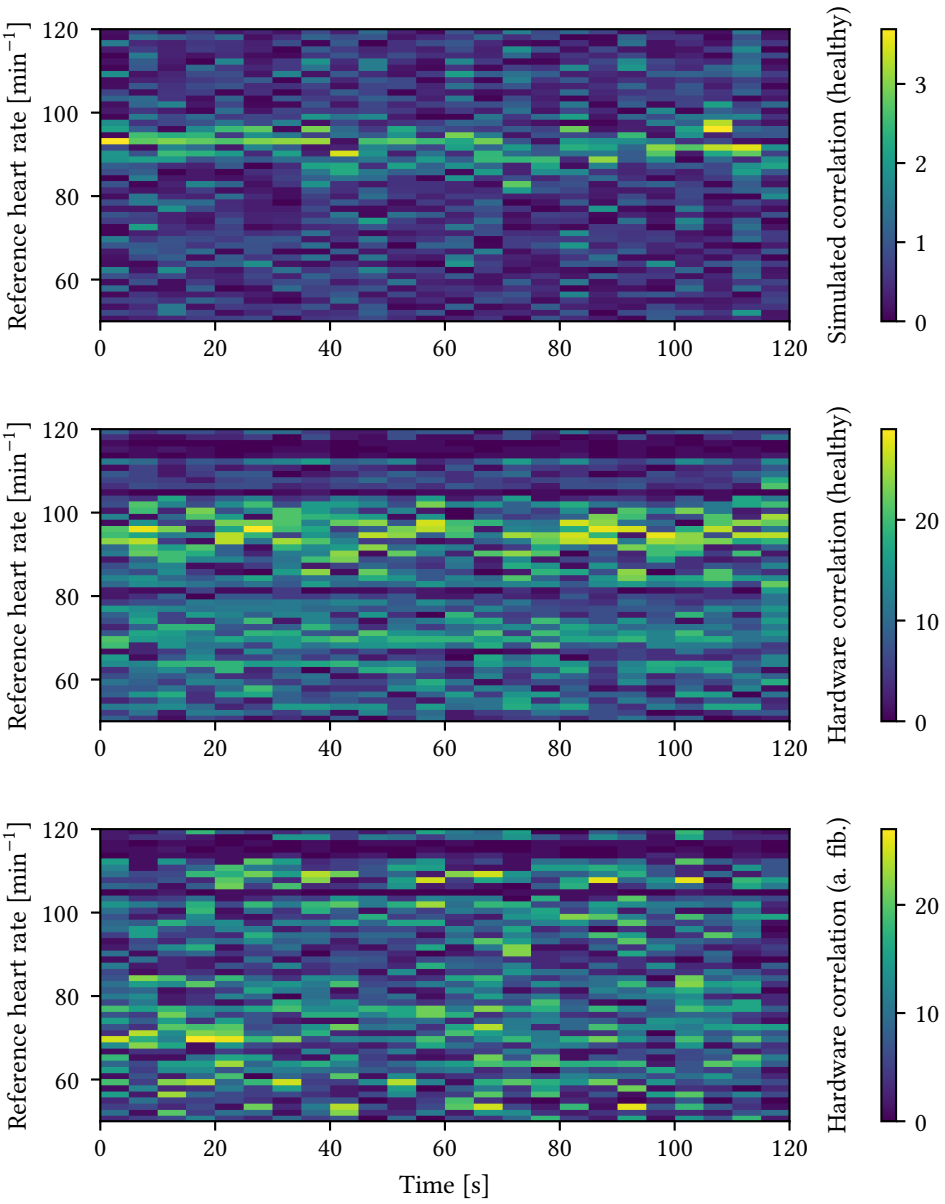


Figure 6.1: Correlation measurements between an input ECG trace and reference pulses generated by the neurons. The absolute value of the difference between causal and acausal correlation is color-coded over time bins and reference heart rates. **Top**: Correlations when simulated on CPU, using an exponential kernel similar to the setup on the chip. We observe that the input spiketrain, which is a healthy person's heart beat, can be correlated with a reference heart beat of roughly 90 min^{-1}. **Middle**: Correlations measured on HICANN-X v1 for the same ECG trace. We see again a higher correlation at roughly 90 min^{-1}, but the area is broader and the image generally appears more noisy. **Bottom**: Correlations for an ECG trace with atrial fibrillation as input, also measured on hardware. We no longer observe a track of high correlation, the distribution is more random. The measurements were taken on chip 09 (setup 69, HICANN-X v1).

the correlations are read out and reset again. The readout and reset requires some 2 µs of time and should thus not increase the estimated runtime too much, as 50 µs would be required to process the spikes of one batch. However, the readout needs to happen for each row of synapses independently, if multiple ECG traces are processed in parallel. If we use the 256 available synapse rows to correlate 256 traces with the neurons, the reset and readout time will be the limiting factor.

I used 48 neurons in total to generate the reference pulses, with frequencies chosen to represent heart rates of 50, 53, 56, ..., 119 min^{-1}. Two neurons were used for each frequency in order to mitigate problems with slight deviations in the frequency and jitter of the spikes. The time constant of the correlation measurement was approximately 10 µs, but not calibrated precisely. To visualize the correlations, I plot the different reference pulse rates along the vertical axis and the time of the incoming ECG signal on the horizontal axis, in bins of 5 seconds. The color-code shows the absolute value of the difference between the causal and acausal correlation measurements. We should be able to see a track of high correlation at one well-matching reference frequency, covering multiple time bins, if the person is healthy and the heartbeat is regular. For atrial fibrillation, we expect a more random distribution of amplitudes. Note that since we only expect sensible signals for healthy persons, the choice of reference pulses only needs to cover healthy heart rates.

The method is initially tested in simulation, by just using NumPy [van der Walt, Colbert, and Varoquaux, 2011] to calculate the exponentially weighted correlations. This shows the method works as expected and a higher correlation between the ECG signal and reference pulses can be observed at a certain frequency for multiple time bins. These simulated differences between causal and acausal correlation amplitudes are shown in the top plot of figure 6.1 for a healthy person. We clearly see that heart rates around 90 min^{-1} match the incoming signal quite well, and continue to do so for all time bins, as we would expect for a healthy person. Executing the same experiment with the same input trace on the chip produces the middle plot in figure 6.1. We observe a much broader trace of correlation, as neurons' spike frequencies are subject to deviations, but generally we can still see the same image as for the simulation, with heart rates of 90 min^{-1} to 100 min^{-1} matching well for most time bins. Finally, we look at a person with atrial fibrillation on hardware. Using this different input spiketrain, the lowest plot in figure 6.1 is generated. The correlations are distributed more randomly, the track of well-matching heart rates has vanished. Hence, the method seems to work on hardware as expected.

For the classification of the obtained correlation data, a simple artificial neural network is used. It consists of two dense layers and has 128 hidden neurons. There was no optimization done, it is well possible that a much smaller network would have achieved equal results. Indeed, this network is very large in order to be integrated on the chip. As this only a proof of concept, the evaluation was only done on the host computer. Using the simulation data, accuracies of 90 % were reached using the mentioned network. Using the results from hardware, the reached accuracy was 88 %, or more precisely: 89 % of the atrial fibrillation traces and 87 % of the sinus rhythms were classified correctly. Porting the final classification to the chip and shrinking the network may decrease accuracy a bit.

In both cases, 14 000 ECG traces were used for training of the network, and 2000 traces were held out in a test set, on which the mentioned accuracy was reached.

Compared to the 91 % accuracy using the Poincaré plot, this method doesn't perform too well. I suspect the largest drawback to be the alignment of neurons' reference spikes to the incoming spikes. For each spike event, the causal and the acausal correlation is stored. In order to create a significant amplitude difference between the two, the time difference between the pre- and postsynaptic spikes has to be quite small. For example, a neuron spiking 1 µs after the presynaptic spike will yield a high causal amplitude, much higher than the acausal amplitude resulting from the 7 µs between the neuron's and the presynaptic spike, assuming a regular pulse of 8 µs inter-spike-interval is arriving. However, if the neuron's spike frequency differs only slightly from the heart rate, the timing will quickly shift and either produce strong acausal amplitudes (in case the neuron now spikes before the incoming pulse), or produce causal and acausal amplitudes in equal parts (in case the time difference approaches 4 µs). Maybe the issue could be mitigated using different time bins or changing the time constant of the exponential kernel. On hardware, the slight irregularities in the neurons' reference pulses, but more importantly, the less precise tuning to the desired inter-spike-interval broaden the track of high correlation for healthy persons, which to some extent certainly impacts accuracy. Besides that, the fixed-pattern deviations between the individual correlation sensors certainly did not help visualization or analytical analysis, but a neural network classifying the results should learn these deviations.

In terms of energy efficiency, this method profits from processing a large number of traces in parallel. The time required for correlation readout can be reduced by filling up all available neurons, so using as much data per synapse row as possible. Since only 48 neurons are used with this setup, five different traces can be processed per synapse row, as all 6 synapse label bits are available. Further, all 256 synapse rows can be used in parallel as well, allowing for parallel processing of 1280 traces in theory. However, the input event rate will eventually impose a limit. Although the input data is sparse, as only one event per heartbeat is sent, 1280 traces almost precisely saturate the available input rate, assuming a heart rate of $120\,\mathrm{min}^{-1}$ for all traces. But since the heart beats are not uniformly distributed, we should stay significantly below these limits in order to not delay or drop spikes. Slight delays of some FPGA clock cycles, i. e. some 10 ns should not be dramatic, as that would only be 1 ms in the time domain of the ECG trace. But larger delays would severely impact the result, as the timing of the incoming spikes is all the information the correlation sensors process. Occasionally dropping a spike altogether may not even be so bad, since the accumulated correlation amplitudes would not change.

The competition's regulations specify a batch size of 500 traces anyway, thus the runtime will be estimated for this case. With five traces processed per row, 100 synapse rows need to be used for the correlation measurements, and read out after each time bin. Estimating a time of 2 µs for the correlation reads per row and a processing time of 50 µs for a bin of 5 s of spike inputs, we would need 6 ms to process all traces in full length. This results in 12 µs of runtime per trace, little more than processing two vectors in hagen mode. When also using the full 120 s of data, an artificial neural network will require more

time per trace, at least with the current amount of downsampling. Still, for this approch, the generated correlation data now has to be further processed. Maybe a much smaller network than the one used here would suffice for that, and we would still have a synapse matrix available. However, the generated amount of data is large, filling nine vectors in signed matrix multiplication mode. Thus, without some analytical pre-processing of the correlations, using only a part of it in the simplest case, the final decision-making will significantly improve the overall runtime, adding at least some 10 ms for all 500 traces. But most importantly, reaching a good accuracy with this method seems more difficult than when using convolutions directly. Hence, we discontinued the work on this approach.

6.2 Short term plasticity

Another possibility to classify the ECG traces is using short term synaptic plasticity (STP) to turn the timings of input spikes into amplitude differences. When configured in depressing mode, STP makes the input spikes' amplitudes decrease if the time between them is short, as shown in figure 2.3. If there is a pause in the spiketrain, amplitudes recover and increase in amplitude again. For facilitation, the operation is inverted: amplitudes grow if spikes are close, and get smaller again during pauses.

We connect two STP synapses to a neuron: One with short term depression in excitatory mode and one with short term facilitation in inhibitory mode. For an input spiketrain showing sinus rhythm, i. e. regular inter-spike intervals, the two synapses' amplitudes can be matched such that they cancel and the neuron doesn't spike. But if there is a longer pause in the spiketrain, the depressing amplitudes would recover and get larger, and the facilitating amplitudes would get smaller - hence triggering a spike in the neuron as response to the late input spike. In the opposite case, for two spikes in short succession, there would also be no spike since the facilitating synapse is connected in inhibitory mode. To also indicate shorter intervals, we use a second neuron with the synapses connected with the opposite sign, i. e. the facilitating synapse in excitatory mode and the depressing synapse in inhibitory mode. If there are two spikes in short succession, the depressing amplitude would be smaller and the facilitating amplitude larger, now leading to a spike in this neuron. We therefore have two neurons indicating whether a given inter-spike interval is too large or too small for a given STP configuration.

The matching of STP amplitudes to different spike frequencies is done by using different recovery time constants. We select a high utilization at each spike in order to track only one inter-spike interval and use 16 synapse drivers with 16 different recovery settings to match different spike frequencies. We further need to use different drivers for the depressing and facilitating synapses, doubling the number of necessary drivers to 32. Finally, we double the number of used drivers again in order to receive higher input amplitudes at the neurons, which also helps at smoothing the remaining offsets of amplitudes after calibration. We thus use all 64 synapse drivers connected to one CapMem block for the processing of STP. For this experiment, we also use all 256 neurons on the synapse matrix since we connect populations of eight neurons instead of only one neuron to indicate if a

pair of drivers at a given recovery setting matches the input spiketrain well or is too fast or too slow.

After experiments with artificial spiketrains quickly show that the method is generally working, I use it to process the ECG traces from our dataset. With a bug on HICANN-X v1 limiting the utilization of synaptic efficacy at each spike, successful experiments with ECG traces could only be started on HICANN-X v2. When plotting the spikes received from the chip, the horizontal axis shows time, and the vertical axis shows the different neurons. They are separated into groups of 16 neurons each, which are connected to the same synapse drivers, thus all processing STP with the same recovery time constant. Red points show spikes of a neuron which reacts to depressing inputs, blue points show reactions to facilitating inputs. This means that for faster input spikes, we have more blue points, and more red points for slower inputs. The populations in the lower part of the plot are configured with a slower recovery, where facilitation dominates more easily. Towards the top of the plot, recovery gets faster, with depressing amplitudes taking over eventually. If an input spike train makes depressing and facilitating amplitudes match very well, the neurons in a group may not spike at all.

Figure 6.2 shows the ECG trace of a healthy person analyzed with the described STP mechanism. We can observe that initially, the person's heart rate is higher, and relaxes with time. The populations gradually change their judgement on the rate, with faster recovery initially matching better, and slower recovery matching near the end of the trace. Apart from the one population which fits the input heart rate well at a given time, all populations respond to each input spike. The populations at faster recoveries indicate the spiking is too slow, and the slower recoveries indicating the spiking is too fast, so everything works as expected. Apart from some single neurons which spike more frequently than others, some even continuing to spike after the input trace has ended, the results look very good.

The same analysis for a person with atrial fibrillation is shown in figure 6.3. We first notice the irregularities in the spiketrains of all the populations, resulting from the irregular input spikes. But importantly, many populations frequently change the spike response between depressing and facilitating, most visible at recovery settings of 6 LSB to 9 LSB. After a pause, the red (depressing) populations fire, during faster bursts, the blue (facilitating) populations fire. The fact that the firing populations rapidly change across multiple STP recovery settings hints at atrial fibrillation. Judging by these plots, the STP method works very well to filter for certain spike frequencies. Since most relevant responses are in the middle range of recovery settings, the further processing will only consider populations at recovery settings 1 LSB to 12 LSB.

In order to obtain a label for each trace, the populations' spikes need to be analyzed. While a second spiking layer, reacting to frequent changes in the active population, could be a nice solution, I will focus on the more traditional approach here. Each population's spikes are first binned in slices of 5 s. On hardware, the binned spikes are directly available using the neurons' spike counters. I will compare a direct analysis and an artificial neural network to obtain the final decision.

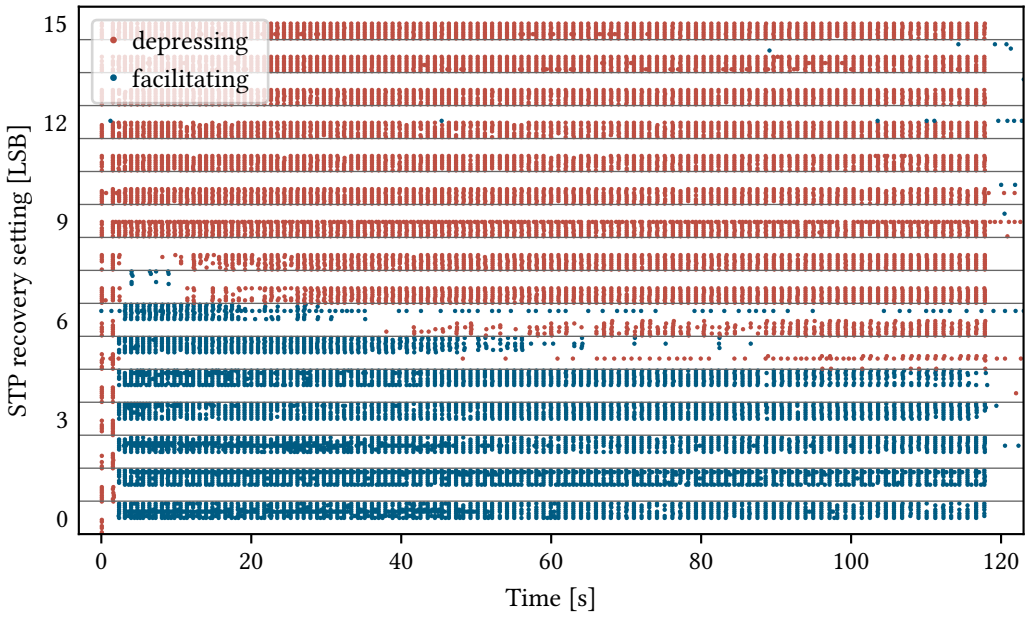


Figure 6.2: STP spike responses when processing an ECG trace showing sinus rhythm using STP. Time is plotted to the right, different populations of neurons stimulated by STP with different recovery time constants are stacked vertically. The populations spike in response to each input spike, extracted as the heartbeat from the ECG trace. The heart rate is higher initially and decreases slowly. STP amplitudes behave differently to the changing input spike rate: For an STP recovery setting of 6 LSB, facilitation dominates initially, at the higher heart rate. As the spike frequency decreases, the amplitudes get more recovered, thus facilitating amplitudes decrease and depressing amplitudes increase. Hence, depression dominates for the later part. Using multiple synapse drivers at different recovery settings, we can evidence regular heart beat as the dominating populations change only slowly. The experiment was run on chip 23 (setup 66, HICANN-X v2).

A simple analytical approach is to consider the product of depressing and facilitating spike counts for each recovery setting. If both the depressing and facilitating populations spike, the product is large, but it is zero if only one of the population spikes. A trace is labelled with atrial fibrillation if too many of these products exceed a threshold. With this method, I obtain an accuracy of 87 %, which was biased such that the competition's requirements are met, with 92 % detection rate for atrial fibrillation and 82 % accuracy for sinus rhythm. While the accuracy is not amazing, it fulfils the requirement and it does not drop any further when everything is executed on hardware, since the multiplication and comparison operations are simple enough to be processed by the PPU, the on-chip microprocessor. The runtime can be reduced by not considering the whole trace. Here, it was tried to process only 50 s of the available 120 s of data, which still yields enough

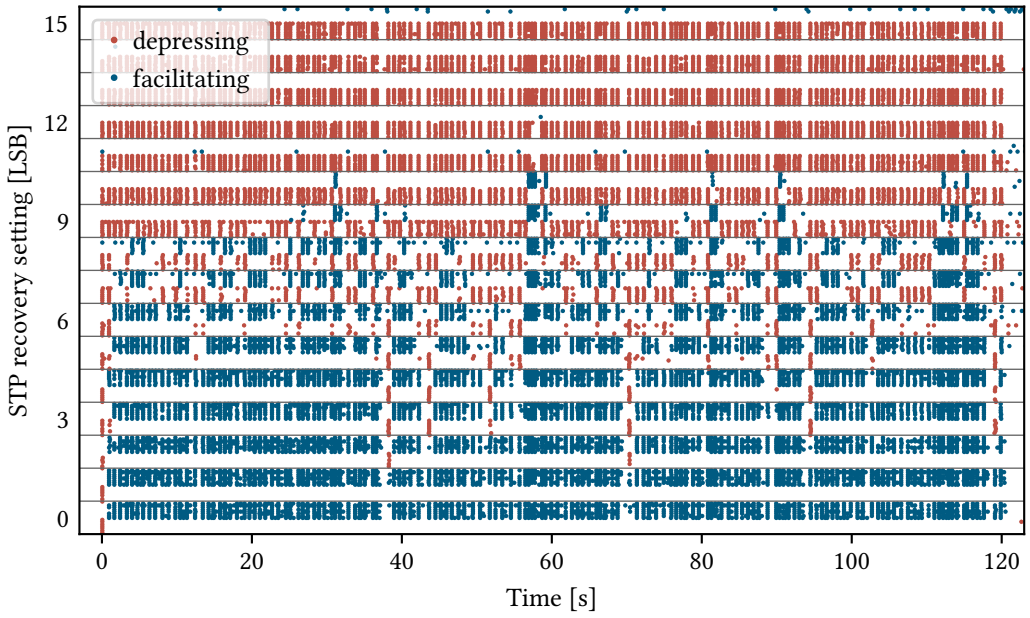


Figure 6.3: STP spike responses when processing an ECG trace showing atrial fibrillation. Time is plotted to the right, different populations of neurons stimulated by STP with different recovery time constants are stacked vertically. The spikes of even the top populations are irregular, following the irregularities in the input trace. The bursts of faster input spikes and the pauses in between make the dominating population shift between depressing and facilitating frequently for multiple STP recovery settings. Pauses are long enough that even at a slow recovery time constant setting of 3 LSB, recovery had enough time to make depressing amplitudes dominate. Conversely, bursts are fast enough to make facilitating populations spike even at a fast recovery setting of 10 LSB. These fast changes in the dominating populations indicate atrial fibrillation. The experiment was run on chip 23 (setup 66, HICANN-X v2).

accuracy, but even more barely, at 91 % and 81 % for the atrial fibrillation and sinus rhythm labels, respectively.

The second evaluation method utilizes an artificial neural network, consisting of two dense layers. The first layer receives the binned spike counts and maps them to 32 neurons, the second layer uses two neurons for the final labelling. The first layer uses a ReLU activation function, the second layer a softmax. Neither uses a bias. The network is not yet ported to hardware, it is trained and evaluated only on CPU. The available data is again split into a training set of 14 000 traces and a test set of 2000 traces. Using the full 120 s of binned spikes, the network reaches an accuracy of 92 %, with 95 % of the atrial fibrillation and 89 % of sinus rhythm classified correctly. The accuracy significantly exceeds the requirements, thus the network is also evaluated on only a slice of the traces. Using only 25 s of inputs, an accuracy of 91 % is maintained, distributed as 94 % of atrial fibrillation

and 87 % of sinus rhythm correctly labelled. The accuracy is identical to the Poincaré plot, which shows the method works well. However, the accuracy may decrease slightly when the network is evaluated on chip, but the performance drop should not be big, at least if we allocate enough resources.

In terms of hardware resources, the first optimization should be using only one neuron per population, since the largest part of the neurons seems usable. Using only 12 recovery settings, which was shown to suffice, the 256 neurons per synapse matrix could be used to process 10 traces in parallel, with 16 neurons still unused. The synapse drivers do not impose an additional limit since they can process STP independently for all 64 synapse labels. If the PPU handles the final classification, we can process 20 traces in parallel on both synapse matrices. Using the network for the final classification, this number will be lower since we need neurons for the inference. However, considering the barely reached accuracy with the analytical classification, we may need to process more time of each trace, thus using the neural network at a smaller part of the input traces is likely to be the better solution.

When using the two-layer network for the final decision, 25 s of input data suffice for good accuracy. In this configuration, the first layer receives spike-counter inputs from 24 neurons for 5 time bins, which means the input vector is filled with 120 entries. The second layer is small as it receives only 32 inputs from the first layer and requires only two neurons for the labels. The first layer's 120 inputs suggest using one synapse matrix in hagen mode to do the classification. However, we can use the synapse drivers in only one mode, hence we would need to allocate one whole synapse matrix for the actually small network. Processing STP only requires 48 synapse drivers, the number is now reduced as we use only 12 of the 16 recovery settings, and we have 80 drivers left. All of those can now be used for hagen mode.

A proposed more efficient placement is sketched in figure 6.4. There, the first dense layer is split into two parts on both synapse matrices, enabling up to 160 inputs, which would allow processing a sixth time bin of data, if this was found to be beneficial. As the final network also requires neurons, the number of traces processing STP in parallel is reduced from 10 to 9, since 32 neurons for the first dense layer and two more neurons for the final decision need to be placed. Obviously, using two synapse matrices processing 9 traces each is better than using one to process 10 traces. Placing the three parts besides each other in one matrix would work, but would only allow for processing 7 traces on the remaining neurons. Further, while for 120 inputs, the layers could be placed besides each other such that the two synapse labels suffice to distinguish their inputs, this is no longer the case if we used more inputs to the first layer. Finally, there are 4 and 6 neurons left unused on the two synapse matrices, which should still be enough in case some neurons are not working well in the STP layer.

In terms of runtime and thereby energy efficiency, this solution should perform quite well. Here, some numbers are estimated for the case that processing 25 s of each trace is sufficient and using only one neuron for each spiking response in the STP layer works well. With the currently used speed-up of 1×10^5 compared to real time of the input traces, we need 250 µs to process 18 traces. This STP processing ultimately poses the bottleneck, as

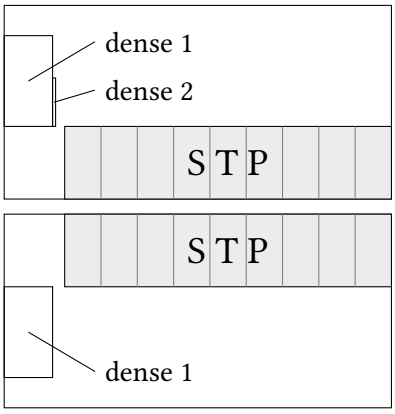


Figure 6.4: Proposed placement of ECG classification layers on chip. The first (STP) layer processes 18 traces in parallel on both synapse matrices. The resulting data is processed by the dense layer 1, which is split across both synapse matrices. The final decision is obtained from the dense layer 2. Both dense layers can operate in parallel and process all 18 datasets generated from the STP layer while this processes the next 18 traces. It should be possible to classify one trace every 5 µs to 10 µs with this setup.

this equals 14 µs processing time per trace. The two hagen-mode layers will only require approximately 5 µs per trace, as they can both be used in parallel.

Future steps to mitigate the current bottleneck, i. e. speeding up the processing of the STP layer, include choosing smaller time constants in the analog circuitry, where I estimate a factor of 2 might still be possible. I have not yet tried to send the inputs faster, since this requires changes at many ends: The STP recovery needs to be configured faster, which is easily possible, but also the neurons' dynamics have to be changed. Mainly, the synaptic input time constants and membrane time constants need to be adjusted, but we will lose signal by reducing the synaptic input time constant. It is to be determined if even at half the time constants, a similar state of easily triggerable spikes can be achieved. Apart from changing the analog dynamics, the number of required STP recovery settings and the number (and size) of time bins can be optimized. The former allows for more parallelism if less neurons are required per trace, the latter simply decreases the amount of data processed. I did not spend a lot of time optimizing these parameters, but I consider it possible that employing both, we will reach a processing time of 5 µs per trace as well.

Ultimately, the question remains at what accuracy the classification can be done. We need to keep in mind that for the good numbers given above, the final network layer has not yet been ported to the chip, and there were still populations of 8 neurons used per recovery setting. When using hagen mode directly, which essentially is using a convolution layer instead of STP, it is unlikely that a processing time of 5 µs is undercut. At this time, the network will not be able to process much more than 5 s of the trace, a trade-off between runtime and accuracy exists in hagen mode as well. But even if both solutions yield enough accuracy at some 5 µs of runtime per trace, the pure hagen mode solution

will have the advantage of requiring less preprocessing in the FPGA, especially if using a smaller slice of trace is sufficient. On the other hand, using spike inputs to the chip is the sparsest coding possible, and each event sent to the chip requires energy as well - but optimizing runtime will be most useful. Regarding the amount of possible and yet untested optimizations for the STP solution, it is too early to profoundly discuss which solution is the best.

7 Discussion and Outlook

BrainScaleS-2 is capable of processing spiking neural networks using analog neurons and synapses following the adaptive exponential integrate-and-fire model. This thesis extends the capabilities for energy-efficient inference to non-spiking artificial neural networks by configuring the chip's analog core for vector-matrix multiplication. A vector of input activations is encoded as pulse widths and multiplied with the synapse weight matrix. Neurons are configured as integrators and accumulate the currents from the synapses. Each row of synapses can connect to either the excitatory or the inhibitory synaptic input of the neurons, allowing for signed weights. The neurons' membrane potentials represent the results of the multiply-accumulate operation and can be digitized in parallel by the CADC. This setup is called hagen mode.

For most of the analog circuitry, calibration routines were developed, countering distortions due to transistor-level mismatch. For hagen mode, we calibrate the synapse drivers, neurons and the CADC. The pulse widths emitted from the synapse drivers could not be calibrated as desired on HICANN-X v1, since the available range of the calibration parameter was insufficient. With a new calibration mechanism on the next chip version, HICANN-X v2, the pulse widths were calibrated better, reaching a standard deviation of approximately 100 ps or 1 LSB of the 5 bit input activations. The CADC calibration worked very well on both versions, the standard deviation between different channels was below 1 LSB or approximately 4 mV.

The neurons were calibrated in an integration mode, with long membrane time constants and short synaptic input time constants. A comparison between the behavior of a typical leaky integrate-and-fire neuron and this integrator setup is shown in figure 3.4. The amplitudes resulting from input spikes recorded on the neurons' membranes were calibrated to a standard deviation of roughly 3 %. The integration follows no dynamics inspired by biology, which allows sending the entries in a vector much quicker than typical spike inputs. For two networks classifying the MNIST dataset of handwritten digits [LeCun and Cortes, 1998], integration times of 2 µs to 3 µs were sufficient on HICANN-X v2. To complete a multiply-accumulate operation however, the initial reset of neurons' membrane potentials to a known starting voltage and the final digitization of the membrane potentials impose additional overheads. Including these, we estimate a vector-matrix multiplication to take 5 µs.

A number of parameters control how the vector-matrix multiplication behaves. Most of them affect the neurons' synaptic input circuitry. We had to set them such that the two main issues, trial-to-trial noise and saturation of synaptic currents, are mitigated. During calibration, the synaptic input OTA's gain and the synaptic input time constant can be selected. Changing the gain did not change a test network's accuracy significantly as both noise and signal are amplified. The synaptic input time constant is much more relevant,

as it controls the decay of the signal at the OTA's input. Hence, it scales the amplitudes, similar to the synapse DAC bias, which we only keep constant at the maximum value. Fundamentally, both these parameters also affect the potential for saturation of the synaptic inputs. But keeping the synaptic input time constant small and the DAC bias large means reaching a stable membrane potential quicker, which allows digitizing it earlier. If we increased the synaptic input time constant, we would also need to increase the wait time between events, which we employ to counter saturation. Another alternative to achieve a stronger signal is repeating all entries in a vector. All approaches increase the integration time, and, at least on HICANN-X v1, where the synaptic input time constants can not be configured as short, the integration time will eventually reach the same order of magnitude as the membrane time constant. Leakage is then no longer negligible as amplitudes decay. An advantage of repeating the vector over choosing a longer synaptic input time constant is that the differences between amplitudes of the first vector entries and those of the last entries are smaller: We first send all vector entries before we start repeating the vector, which means the vector entries are distributed more evenly within the integration time. Hence, we mainly use vector resends to reach a good signal to noise ratio.

Tuning the neuron's membrane capacitance allows using fewer vector sends and reduces the runtime of the multiplication. At a lower capacitance, the synaptic input OTA needs to emit less charge to achieve the same voltage on the membrane. While noise, i. e. random current, is integrated to higher voltages as well, the shorter time from reset to readout means that the amplitude of the noise is smaller, assuming it can be considered a random walk process. But even if the benefit in signal-to-noise ratio is small, the shorter integration time is an advantage in terms of overall runtime and therefore energy efficiency. A smaller membrane capacitance even decreases the time required to reset the potentials, since a smaller membrane time constant is reached at the same conductivity. For classifying MNIST, there was no drop in accuracy observable when using fewer vector sends at a reduced membrane capacitance.

Speeding up the multiplication further would be possible at the cost of precision in the digitization. The 8 bit resolution of the CADC is achieved by counting clock cycles until a ramp voltage exceeds the voltage in a channel. Setting the ramp current, e. g., twice as high would allow making the ramp cross the dynamic range twice as fast, while the counter would only pass the lower 7 bit. We could therefore gain 0.5 µs of runtime per vector by sacrificing one bit of result precision. Selecting a smaller precision of digital results is fully supported during calibration. While we could thus speed up the digitization significantly, the potential for runtime reduction is limited, since the digitization only takes approximately 1 µs currently. The runtime of a MAC operation will eventually be limited by the integration time.

A fundamentally different approach to accumulation is using only the line connecting all synapses, and not the neuron's membrane. Most notably, this avoids using the synaptic input OTA, which is responsible for the noise on the membrane. Each synapse still decreases the potential on this line reaching the input of the OTA, but we only digitize the potential on this line. However, we can not connect it directly to the CADC, since the idle potential on this line is 1.2 V, while the CADC can only digitize voltages up to

approximately 1.1 V. We can use the source follower in the synaptic input circuitry to drop the potential by roughly 0.6 V, bringing it into a suitable range. But this means that only one channel per column of synapses can be used, and we can not have two signs for the weights in one column. We could instead use two synapse columns to represent the signed weights, and calculate the difference in the digitized result. This, while cutting the number of available "neurons" in half, is not such a big issue as this would allow using all rows of synapses, doubling the input vector size to 256 entries. The number of possible multiplications in one synapse matrix is thus constant, but the addition of positive and negative amplitudes is no longer handled by an analog circuit.

While I expect the trial-to-trial noise to be reduced using this approach, the problem with saturation and fixed pattern noise may not be solved. The synaptic DACs begin to saturate for output voltages of approximately 1.0 V and lower. Still, it would be much easier to notice this saturation, as we digitize exactly this potential. Using the neuron as integrator, saturation of the synaptic inputs can happen even when the resulting membrane potentials are lower. However, the dynamic range of voltage is then limited to some 200 mV. This means the synapse DAC bias current needs to be tuned precisely to suitably map the results into this range. Considering the CADC works very well this should at least be sufficient for 6 bit result precision. The analog precision still depends on the calibration of synapse drivers, the fixed-pattern noise of synapses and possible variations in the capacitances of the synapse lines, which we would not be able to calibrate.

The biggest advantage of integrating on the synapse lines is that we would no longer need to wait between events. Since we integrate on this line and do not want to recharge it, the synaptic input time constant would be set long. Integration of a full vector of 256 entries could happen within 1 µs, as we could send two events each FPGA clock cycle. Even with the CADC still needing 0.25 µs for digitization, this method would be significantly faster. The biggest problem of this approach is that there is no convenient way to reset the potentials on the lines. The lines in each synapse column could be switched to a common debug line where we could provide a potential of 1.2 V, but this process may consume some time. It is important to know the precise starting voltage before integration, especially with the lower dynamic range here. If the reset did not clear any previous state and did not yield the same potential across all synapse lines, a CADC baseline read before each vector input would be required, adding even more runtime. Hence, although we never tested the concept, we expect that the method currently is not much faster than the integration on the neurons' membranes.

A more significant speedup of vector-matrix multiplication requires hardware changes. A vector of input activations could be provided by the PPU and written directly to the synapse drivers, instead of relying on events from the FPGA. For the accumulation on the synapse line, we need a quick reset and a sample-and-hold feature for the CADC, allowing to proceed with resets while the digitization is still happening. Ideally, a new circuit would replace all the discussed components and only keep the potential on the synapse line at 1.2 V, while measuring the charge that flows off the line when all the synapses process their events. This charge measurement would be equivalent to the accumulation of charge on the neuron's membrane. Compared to integration on the synapse line, it would

avoid problems with the dependency of synapse currents on the line potential and not require an equal capacitance for all the accumulation lines.

In its current state, with integration on the neurons' membranes, the MAC operation is used to classify MNIST handwritten digits [LeCun and Cortes, 1998], which works well although the mapping of vector entries to pulse widths is not perfectly linear (figure 4.5): Even without training on hardware, an accuracy of approximately 92 % was reached on the 10 000 test images. After training with hardware in the loop, the accuracy of a convolutional network was improved to 98 %, only 0.4 % lower than the same model's accuracy on CPU after training at similar weight precision.

The main motivation to use HICANN-X for matrix multiplication is its low energy consumption. A smaller MNIST network, consisting of two dense layers with 64 hidden neurons, can process one full-size image every 10 µs, assuming batched execution. The chip is estimated to use only 0.36 W of power during inference, requiring 3.6 µJ per image. This network reached an accuracy of (96.30 ± 0.09) % after training on hardware. A comparable spiking network on the chip, using 118 hidden neurons, can process one down-sampled image every 14 µs at almost identical accuracy [Cramer et al., 2020]. Also the energy consumption is very similar: The sparser input coding in the spiking network allows some communication links between FPGA and chip to be shut down, decreasing power consumption. Hence, the spiking solution uses 4 µJ per image despite the larger runtime.

Compared to an efficient digital solution, Google's Edge TPU, which reaches 2 TOPS/W [Coral Edge TPU, 2020], our solution is inferior at an estimated 0.15 TOPS/W. Still, the combination of hagen-mode to process large amounts of input data and spiking subsequent layers could be an efficient solution. This is especially true if we implement some of the hardware changes mentioned before and increase the throughput in hagen mode.

For the more complex ECG classification task, distinguishing sinus rhythm from atrial fibrillation, different approaches were investigated. Besides a deep convolutional network in hagen mode, the synapses' correlation sensors or their short term plasticity feature can be used to generate data, which again needs to be processed to arrive at a label. Especially using short term plasticity as a filter for the inter-spike-intervals of the incoming trace and a small artificial neural network afterwards yields good accuracy at a potentially short runtime of 5 µs to 10 µs per trace. The method classified 94 % of atrial fibrillation and 87 % of sinus rhythm correctly, exceeding the competition's requirements. While the short term plasticity layer was executed on chip, optimizations required to reach the mentioned runtime were not yet implemented and the final network was only processed on CPU. It is well possible that the accuracy decreases when implementing the proposed solution on chip. A pure hagen mode solution could perform equally well when considering both accuracy and runtime. If the approaches do not differ significantly, the non-spiking input for hagen mode will be advantageous, as it requires less preprocessing.

Bibliography

Aamir, S. A., P. Müller, G. Kiene, L. Kriener, Y. Stradmann, A. Grübl, J. Schemmel, and K. Meier (Oct. 2018). „A Mixed-Signal Structured AdEx Neuron for Accelerated Neuromorphic Cores“. In: *IEEE Transactions on Biomedical Circuits and Systems* 12.5, pp. 1027–1037. ISSN: 1932-4545. DOI: 10.1109/TBCAS.2018.2848203.

Aamir, S. A., P. Müller, L. Kriener, G. Kiene, J. Schemmel, and K. Meier (Oct. 2017). „From LIF to AdEx Neuron Models: Accelerated Analog 65 nm CMOS Implementation“. In: *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, pp. 1–4. DOI: 10.1109/BIOCAS.2017.8325167.

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. URL: http://download.tensorflow.org/paper/whitepaper2015.pdf.

Ambrogio, Stefano, Pritish Narayanan, Hsinyu Tsai, Robert M Shelby, Irem Boybat, Carmelo di Nolfo, Severin Sidler, Massimo Giordano, Martina Bodini, Nathan C.P. Farinha, et al. (2018). „Equivalent-accuracy accelerated neural-network training using analogue memory“. In: *Nature* 558.7708, pp. 60–67. DOI: 10.1038/s41586-018-0180-5.

Anan, Tsuyoshi, Kenji Sunagawa, Haruo Araki, and Motoomi Nakamura (1990). „Arrhythmia analysis by successive RR plotting“. In: *Journal of Electrocardiology* 23.3, pp. 243–248. ISSN: 0022-0736. DOI: 10.1016/0022-0736(90)90163-V.

Bianchini, Monica and Franco Scarselli (2014). „On the complexity of neural network classifiers: A comparison between shallow and deep architectures“. In: *IEEE transactions on neural networks and learning systems* 25.8, pp. 1553–1565. DOI: 10.1109/TNNLS.2013.2293637.

Billaudelle, Sebastian (2017). „Design and Implementation of a Short Term Plasticity Circuit for a 65 nm Neuromorphic Hardware System“. Master's thesis. Universität Heidelberg.

Billaudelle, Sebastian, Yannik Stradmann, Korbinian Schreiber, Benjamin Cramer, Andreas Baumbach, Dominik Dold, Julian Göltz, Akos F. Kungl, Timo C. Wunderlich, Andreas Hartel, Eric Müller, Oliver Breitwieser, Christian Mauch, Mitja Kleider, Andreas Grübl, David Stöckel, Christian Pehle, Arthur Heimbrecht, Philipp Spilger, Gerd Kiene, Vitali Karasenko, Walter Senn, Mihai A. Petrovici, Johannes Schemmel, and Karlheinz Meier

(2019). *Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate*. arXiv: 1912.12980 [q-bio.NC].

Boser, B. E., E. Sackinger, J. Bromley, Y. Le Cun, and L. D. Jackel (Dec. 1991). „An Analog Neural Network Processor with Programmable Topology." In: *IEEE Journal of Solid-State Circuits* 26, pp. 2017–2025. DOI: 10.1109/4.104196.

Brette, Romain and Wulfram Gerstner (2005). „Adaptive exponential integrate-and-fire model as an effective description of neuronal activity". In: *Journal of neurophysiology* 94.5, pp. 3637–3642. DOI: 10.1152/jn.00686.2005.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].

Bruun, E. and P. Shah (1995). „Dynamic range of low-voltage cascode current mirrors". In: *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*. Vol. 2, pp. 1328–1331. DOI: 10.1109/ISCAS.1995.520391.

Bundesministerium für Bildung und Forschung, Federal Ministry of Education and Research, Germany (Mar. 2019). *Richtlinie zur Förderung des Pilotinnovationswettbewerbs "Energieeffizientes KI-System"*. URL: https://www.bmbf.de/foerderungen/bekanntmachung-2371.html.

Coral (2020). *Google Edge TPU performance benchmarks*. URL: https://coral.ai/docs/edgetpu/benchmarks/ (visited on 08/15/2020).

Cosemans, S, B Verhoef, J Doevenspeck, IA Papistas, F Catthoor, P Debacker, A Mallik, and D Verkest (2019). „Towards 10000TOPS/W DNN Inference with Analog in-Memory Computing–A Circuit Blueprint, Device Options and Requirements". In: *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, pp. 22.2.1–22.2.4. DOI: 10.1109/IEDM19573.2019.8993599.

Cramer, Benjamin, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Johannes Schemmel, and Friedemann Zenke (2020). „Training spiking multi-layer networks with surrogate gradients on an analog neuromorphic substrate". In: arXiv: 2006.07239 [cs.NE].

Draghici, Sorin (2002). „On the capabilities of neural networks using limited precision weights". In: *Neural Networks* 15.3, pp. 395–414. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(02)00032-1.

Friedmann, S., J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier (2017). „Demonstrating Hybrid Learning in a Flexible Neuromorphic Hardware System". In: *IEEE Transactions on Biomedical Circuits and Systems* 11.1, pp. 128–142. ISSN: 1932-4545. DOI: 10.1109/TBCAS.2016.2579164.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge (2015). *A Neural Algorithm of Artistic Style*. arXiv: 1508.06576 [cs.CV].

Glorot, Xavier and Yoshua Bengio (May 2010). „Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

Go, Alan S., Elaine M. Hylek, Kathleen A. Phillips, Yuchiao Chang, Lori E. Henault, Joe V. Selby, and Daniel E. Singer (2001). „Prevalence of diagnosed atrial fibrillation in adults: national implications for rhythm management and stroke prevention: the AnTicoagulation and Risk Factors in Atrial Fibrillation (ATRIA) Study". In: *Jama* 285.18, pp. 2370–2375. DOI: 10.1001/jama.285.18.2370.

Gutierrez, Cecilia and Daniel G. Blanchard (2016). „Diagnosis and treatment of atrial fibrillation". In: *American Family Physician* 94.6, pp. 442–452.

Hannun, Awni Y., Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia, and Andrew Y. Ng (2019). „Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network". In: *Nature medicine* 25.1, p. 65. DOI: 10.1038/s41591-018-0268-3.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (June 2016). „Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Hebb, Donald O. (1949). *The Organization of Behaviour*. New York: Wiley.

Hering, H. E. (1908). „Über den Pulsus irregularis perpetuus [Electrocardiogram of permanent irregular pulse]". In: *Deutsch. Arch. f. klin. Med* 94, p. 185.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). „Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hock, Matthias (July 2014). „Modern semiconductor technologies for neuromorphic hardware". PhD thesis. Universität Heidelberg.

Hock, Matthias, Andreas Hartel, Johannes Schemmel, and Karlheinz Meier (2013). „An analog dynamic memory array for neuromorphic hardware". In: *2013 European Conference on Circuit Theory and Design (ECCTD)*. IEEE, pp. 1–4. DOI: 10.1109/ECCTD.2013.6662229.

Hu, Yu Hen, Willis J. Tompkins, Jose L. Urrusti, and Valtino X. Afonso (1993). „Applications of artificial neural networks for ECG signal detection and classification". In: *Journal of electrocardiology* 26 Suppl, pp. 66–73. ISSN: 0022-0736. URL: http://europepmc.org/abstract/MED/8189150.

Intel (2013). *Intel® Core™ i7-4771 Processor*. URL: https://ark.intel.com/content/www/us/en/ark/products/77656/intel-core-i7-4771-processor-8m-cache-up-to-3-90-ghz.html (visited on 08/21/2020).

Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. (2017). „In-datacenter performance analysis of a tensor processing unit“. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12. DOI: 10.1145/3079856.3080246.

Kiene, Gerd (2017). „Mixed-Signal Neuron and Readout Circuits for a Neuromorphic System“. Master's thesis. Universität Heidelberg.

Kim, W., R.L. Bruce, T. Masuda, G.W. Fraczak, N. Gong, P. Adusumilli, S. Ambrogio, H. Tsai, J. Bruley, J.-P. Han, et al. (2019). „Confined PCM-based analog synaptic devices offering low resistance-drift and 1000 programmable states for deep learning“. In: *2019 Symposium on VLSI Technology*. IEEE, T66–T67. DOI: 10.23919/VLSIT.2019.8776551.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). „Deep learning“. In: *nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539.

LeCun, Yann and Corinna Cortes (1998). *The MNIST database of handwritten digits*. URL: http://yann.lecun.com/exdb/mnist/.

Leibfried, Aron (2018). „On-chip calibration of analog neuromorphic circuits“. Bachelor's thesis. Universität Heidelberg.

Lewis, Thomas (1909). „Report CXIX. Auricular fibrillation: a common clinical condition“. In: *British medical journal* 2.2552, p. 1528. DOI: 10.1136/bmj.2.2552.1528.

Micikevicius, Paulius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. (2017). *Mixed precision training*. arXiv: 1710.03740 [cs.AI].

Müller, Eric, Christian Mauch, Philipp Spilger, Oliver Julien Breitwieser, Johann Klähn, David Stöckel, Timo Wunderlich, and Johannes Schemmel (Mar. 2020). „Extending BrainScaleS OS for BrainScaleS-2“. In: arXiv: 2003.13750 [cs.NE].

Naud, Richard, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner (Nov. 2008). „Firing patterns in the adaptive exponential integrate-and-fire model“. In: *Biological Cybernetics* 99.4, pp. 335–347. DOI: 10.1007/s00422-008-0264-7.

Oh, Kyoung-Su and Keechul Jung (2004). „GPU implementation of neural networks“. In: *Pattern Recognition* 37.6, pp. 1311–1314. DOI: 10.1016/j.patcog.2004.01.013.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Rosenblatt, Frank (1958). „The perceptron: a probabilistic model for information storage and organization in the brain.“ In: *Psychological review* 65.6, p. 386. DOI: 10.1037/h0042519.

Rothberger, CJ. and H. Winterberg (1909). „Vorhofflimmern und arrhythmia perpetua [Atrial fibrillation and permanent arrhythmia]“. In: *Wien Klin Wochenschr* 22, pp. 839–844.

Sarkar, S., D. Ritscher, and R. Mehra (2008). „A Detector for a Chronic Implantable Atrial Tachyarrhythmia Monitor“. In: *IEEE Transactions on Biomedical Engineering* 55.3. DOI: 10.1109/TBME.2007.903707.

Schemmel, Johannes, Sebastian Billaudelle, Philipp Dauer, and Johannes Weis (2020). *Accelerated Analog Neuromorphic Computing*. arXiv: 2003.11996 [cs.NE].

Schemmel, Johannes, Steffen Hohmann, Karlheinz Meier, and Felix Schürmann (2004). „A mixed-mode analog neural network using current-steering synapses“. In: *Analog Integrated Circuits and Signal Processing* 38.2-3, pp. 233–244. DOI: 10.1023/B:ALOG.0000011170.92377.6e.

Schmidhuber, Jürgen (Jan. 2015). „Deep learning in neural networks: An overview“. In: *Neural Networks* 61, pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.

Schmitt, Sebastian, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Güttler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, Vitali Karasenko, Mitja Kleider, Christoph Koke, Alexander Kononov, Christian Mauch, Eric Müller, Paul Müller, Johannes Partzsch, Mihai A. Petrovici, Bernhard Vogginger, Stefan Schiefer, Stefan Scholze, Vasilis Thanasoulis, Johannes Schemmel, Robert Legenstein, Wolfgang Maass, Christian Mayr, and Karlheinz Meier (2017). „Classification With Deep Neural Networks on an Accelerated Analog Neuromorphic System“. In: *Proceedings of the 2017 IEEE International Joint Conference on Neural Networks*. DOI: 10.1109/IJCNN.2017.7966125.

Schreiber, Korbinian (2020). „Accelerated Neuromorphic Agents“. (In preparation). PhD thesis. Universität Heidelberg.

Schwartz, Roy, Jesse Dodge, Noah A. Smith, and Oren Etzioni (2019). *Green AI*. arXiv: 1907.10597 [cs.CY].

Simard, Patrice Y, David Steinkraus, and John C Platt (2003). „Best practices for convolutional neural networks applied to visual document analysis.“ In: *International Conference on Document Analysis and Recognition*. Vol. 3. DOI: 10.1109/ICDAR.2003.1227801.

So, David R., Chen Liang, and Quoc V. Le (2019). *The Evolved Transformer*. arXiv: 1901.11117 [cs.LG].

Spilger, Philipp, Eric Müller, Arne Emmel, Aron Leibfried, Christian Mauch, Christian Pehle, Johannes Weis, Oliver Breitwieser, Sebastian Billaudelle, Sebastian Schmitt, Timo C. Wunderlich, Yannik Stradmann, and Johannes Schemmel (June 2020). „hxtorch: PyTorch for BrainScaleS-2 — Perceptrons on Analog Neuromorphic Hardware“. In: *Proceedings of the Workshop on IoT, Edge, and Mobile for Embedded Machine Learning (ITEM)/ECML-PKDD 2020*. arXiv: 2006.13138 [cs.NE].

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2019). *Energy and Policy Considerations for Deep Learning in NLP*. arXiv: 1906.02243 [cs.CL].

Thompson, Neil C., Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso (2020). *The Computational Limits of Deep Learning*. arXiv: 2007.05558 [cs.LG].

Tsodyks, M. and H. Markram (1997). „The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability". In: *Proceedings of the National Academy of Sciences USA* 94, pp. 719–723. DOI: 10.1073/pnas.94.2.719.

van der Walt, S., S. C. Colbert, and G. Varoquaux (Mar. 2011). „The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2, pp. 22–30. ISSN: 1558-366X. DOI: 10.1109/MCSE.2011.37.

Verleysen, M., P. Thissen, J. -. Voz, and J. Madrenas (1994). „An analog processor architecture for a neural network classifier". In: *IEEE Micro* 14.3, pp. 16–28. DOI: 10.1109/40.285221.

Weis, Johannes, Philipp Spilger, Sebastian Billaudelle, Yannik Stradmann, Arne Emmel, Eric Müller, Oliver Breitwieser, Andreas Grübl, Joscha Ilmberger, Vitali Karasenko, Mitja Kleider, Christian Mauch, Korbinian Schreiber, and Johannes Schemmel (June 2020). „Inference with Artificial Neural Networks on Analog Neuromorphic Hardware". In: *Proceedings of the Workshop on IoT, Edge, and Mobile for Embedded Machine Learning (ITEM)/ECML-PKDD 2020*. arXiv: 2006.13177 [cs.NE].

Wu, Zhanghao, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han (2020). *Lite Transformer with Long-Short Range Attention*. arXiv: 2004.11886 [cs.CL].

Wunderlich, Timo, Akos F. Kungl, Eric Müller, Andreas Hartel, Yannik Stradmann, Syed Ahmed Aamir, Andreas Grübl, Arthur Heimbrecht, Korbinian Schreiber, David Stöckel, Christian Pehle, Sebastian Billaudelle, Gerd Kiene, Christian Mauch, Johannes Schemmel, Karlheinz Meier, and Mihai A. Petrovici (2019). „Demonstrating Advantages of Neuromorphic Computation: A Pilot Study". In: *Frontiers in Neuroscience* 13, p. 260. ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00260.

Yamaguchi, Masatoshi, Goki Iwamoto, Hakaru Tamukoh, and Takashi Morie (2019). *An Energy-efficient Time-domain Analog VLSI Neural Network Processor Based on a Pulse-width Modulation Approach*. arXiv: 1902.07707 [cs.ET].

Yin, Shouyi, Peng Ouyang, Shibin Tang, Fengbin Tu, Xiudong Li, Leibo Liu, and Shaojun Wei (2017). „A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications". In: *2017 Symposium on VLSI Circuits*. IEEE, pp. C26–C27. DOI: 10.23919/VLSIC.2017.8008534.

Zucker, Robert S. and Wade G. Regehr (2002). „Short-term synaptic plasticity". In: *Annual Review of Physiology* 64. PMID: 11826273, pp. 355–405. DOI: 10.1146/annurev.physiol.64.092501.114547.

The work carried out in this Master's Thesis used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 785907 and 945539 (Human Brain Project, HBP).

Statement of Originality (Erklärung)

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, January 25, 2021

.....
(signature)