Full Spikey Chip Characterisation and Implementation of RBM-Hidden Layer for MNIST-Classification

Project Report: Carola Fischer, August 17, 2018

ABSTRACT - For MNIST Classification of 12x12pixel binarized MNIST-Numbers 0,1,2,3,4 and 7, the full Spikey has been analyzed. Activation Functions have been measured, hardware parameters were crosschecked and the new image set (100:20, training images:test images) was tested against hardware problems. At last, the hidden layer of a hierarchical structure was implemented and the test set classified.

1 Introduction

1.1 Motivation

The interest in artificial neural networks increased in the past years. They can learn to fulfill complex tasks, like the reduction from lots of input to very few conclusions. This allows a broad field of possible applications from medicine over robotics to data classification. One specific network and application are Restricted Boltzmann Machines (RBM) to classify patterns. [8]
Even though ANNs show promising results, software realization of huge networks is expensive, because computers are not made for strongly parallelized processes. Inspired by this, neuromorphic hardware aims to solve this problem, because it increases the speed up to the order of 10^3 to 10^4. [3, 6]
This report describes how biologically motivated LIF neurons on the neuromorphic hardware Spikey are able to classify the MNIST dataset with a hierarchical feed-forward structure. This was already implemented and tested in [7], though it only used half of the Chip's resources. The task is to characterize the full chip and test for improvement of the classfication task. Hypothetically an increase in better neurons should increase the classification success.
At first an overview over the neuron model, sampling and RBMs is given. Then the Spikey environment is analyzed to test for potential problems. At last the structure is implemented and classifies a subset of handwritten digits.

1.2 Neuron Model

For information flow in biological neurons the cell membrane plays a crucial role with its active and passive ion channels. The Leaky-Integrate-And-Fire model (LIF) builts a simplified picture of the membrane as a capacitor C_m. Its time dependent potential V describes the neuron's behavior. [6]

$$\begin{aligned} C_\mathrm{m}\frac{dV}{dt} =& g_\mathrm{l}(V - V_\mathrm{rest}) \\ &+ \sum_j g_j(t)(V - E_\mathrm{x}) \\ &+ \sum_k g_k(t)(V - E_\mathrm{i}) \end{aligned} \tag{1}$$

The first term accounts for passive ion channels, where g_l is the leak conductance and E_l the resting or leak potential. If no other input exists, the neuron will exponentially decay towards this resting point with the time constant $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$. For information to flow neurons have to be connected. Those links are synapses that can either be inhibitory, hence depress neuronal activity, or excitatory. In both cases an absolute maximum E_k, E_i is given, called reversal potential and the connection strength of each synapse is given by the conductances g_j, g_k. This synapse model is called COBA -Model ("conductance based").

In reality neurons transfer information with so called action potentials (AP), which describes a spike in the membrane potential, followed by a regenerating period where it cannot release an AP again. It is refractory.

$$\text{If} \quad V(t_\mathrm{spike}) \geq V_\mathrm{thresh} \quad \Rightarrow \quad V(t) = V_\mathrm{reset} \quad \text{for} \quad t \in [t_\mathrm{spike}, t_\mathrm{spike} + \tau_\mathrm{ref}] \tag{2}$$

The LIF neuron implements spikes with a threshold V_th. If the membrane potential surpasses the threshold, V is set back to a reset potential V_reset for a fixed time τ_ref, the refractory period. Afterwards it is free to react freely to any input it receives.

The AP is transferred to connected neurons and changes depending on the synapse's nature the postsynaptic potential (PSP). The form and amplitude of a PSP can vary in biology. In this COBA model, we work with exponential PSP kernels (3), which are implemented through the time dependent conductances $g_{j,k}$ from (1).

$$g_{j,k} = w_\mathrm{syn} \sum e^{-\frac{t - t_\mathrm{spike}}{\tau_\mathrm{s}}} * \theta(t - t_\mathrm{spike}) \tag{3}$$

In this case θ is the Heaviside-Function, and defines the starting point of the PSP. The strength or amplitude is given by the synaptic weight w_s. The PSP decays with the time constant τ_s.

For further use we translate equation 1 into the much shortened version:

$$\tau_\mathrm{eff}(t)\frac{dV}{dT} = V_\mathrm{eff}(t) - V(t) \tag{4}$$

with some substitutions:

$$g_{\text{tot}}(t) = \sum_i g_i(t) \qquad \tau_{\text{eff}} = \frac{C_{\text{m}}}{g_{\text{tot}}} \qquad V_{\text{eff}} = \frac{\sum_i g_i E_i}{g_{\text{tot}}} \tag{5}$$

Those LIF Neurons are implemented on the Spikey Chip. Fig. 1 shows a schematic of the basic structures. The Neuron (C) follows equation 1 and

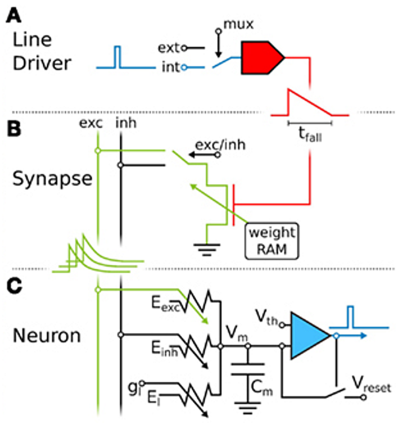

Figure 1: Schematic of a LIF Neuron on Spikey (from [3])

2. The spike is released as a logic pulse that is converted into a voltage ramp by the Line Driver (A). Amongst rise time, fall time and amplitude, the Liner Driver also determines whether the connected row of synapses are excitatory or inhibitory. The Synapse (B) turns the voltage ramp into an weighted exponential that is fed into the connected neuron. The weight has a 4 bit resolution and can be adjusted individually. Other parameters are global. Those include threshold, resting, reset and reversal potential and the refractory period. Latter is controlled by a current $I_{\text{cb}} \in [0, 2.5]$. Small I_{cb} correspond to large τ_{ref}.

All together the Chip consists of two halves with 192 neurons and 256 line drivers.[3, 6]

1.3 Neural Sampling

Under certain conditions we can sample from binary nonzero probability distributions with a network of the previously described neurons, especially from a Boltzmann (BM) distribution $p(\mathbf{z})$:

$$p(\mathbf{z}) = \frac{1}{Z} \exp[-E(\mathbf{z})] \tag{6}$$

where the random state $\mathbf{z} \in \{0; 1\}^N$ is created from a network with N binary random variables. To norm the distribution (6), we divide it with its partition function $Z = \sum_{\mathbf{z}} \exp[-E(\mathbf{z})]$.
The distribution assigns an energy function to every random state. It describes the connectivity between the random variables, the weight $w \in \mathbb{R}^{N \times N}$, and adds a bias (energy offset) $B \in \mathbb{R}^N$ to each variable.

$$E(\mathbf{z}) = -\frac{1}{2} \sum_{i \neq j} w_{ij} z_i z_j - \sum_i b_i z_i \tag{7}$$

Now, we connect the random variable z_k with the neuron's membrane potential by assuming the potential corresponds to the log-odds of variable. For the Boltzmann case, that leads to:

$$u_{\mathrm{k}} = \log(\frac{p(z_k = 1|\mathbf{z}_{\setminus k})}{p(z_k = 0|\mathbf{z}_{\setminus k})}) \quad \xrightarrow[\text{BM}]{} \quad u_k = b_k + \sum_i^N w_{ik} \tag{8}$$

From this follows the conditional probability $p(z_k = 1|\mathbf{z}_{\setminus k})$, that the neuron is in the on-state. The function is a sigmoid.

$$\sigma(u_k) := p(z_k = 1|\mathbf{z}_{\setminus k}) = \frac{1}{1 + \exp(-u_k)} \tag{9}$$

Equation 9 is called the activation function. [1, 5, 6]

This sampling method, called abstract neuron sampling, can approximate any non-zero (??) distribution with enough time steps well. [1] The on-state is defined as the period, where the neuron is refractory. If the true probability distribution is unknown, one can approximate it by sampling. One possible method is the Markov Chain Monte Carlo (MCMC) method that draws the next sample state at time step T+1 from the current state at time step T. It can be shown, that LIF neurons sample from the given distribution under certain circumstances. [2, 6]

At first the LIF neuron itself is deterministic. For sampling stochastic neurons are essential. Therefore we connect the LIF neuron to inhibitory and excitatory background noise sources, in this case poisson noise. If the background is a high frequency source with small weights the neuron will go into the High Conductance State (HCS). It follows that $g_{\mathrm{tot}} \approx \overline{g_{\mathrm{tot}}}$, hence the time dependency of g_{tot} as well as τ_{eff} in equation 4 vanishes. Also τ_{eff} becomes approximately 0. This leaves as an analytically solvable equation. If a spike other than the background occurs the PSP is proportional to the difference of two exponentials.

Second it was shown that the activation function of a LIF neuron in the HCS follows a sigmoid curve, hence we can translate the membrane potential $\overline{u_k}$, but need to translate the LIF domain into a probabilty domain with a scaling factor α.

$$p(z_k = 1|\mathbf{z}_{\setminus k}) = \sigma(\frac{\overline{u_k} - \overline{u_k^0}}{\alpha}) \tag{10}$$

Additionaly the potential is shifted by $\overline{u_k^0}$. $\overline{u_k^0}$ is the value where $\sigma(\overline{u_k^0}) = 0.5$. In this model it was assumed, that $\tau_{\text{eff}} \approx 0$, hence the membrane potential follows the effective potential nearly immediately. If this assumption is not valid, asymmetries in the activation function are observed.

Third, weights and biases of the BM have to be translated into the LIF domain, too. We obtain the bias via the same shift rule as the membrane potential. For the weight we compare the PSPs. In both sampling cases, abstract and LIF, they have to have the same effect even though their shapes differ. Therefore the integral over both PSPs are equal and we can transform one into another.

1.4 Classification with RBMs

A possible implementation to solve classification problems are restricted Boltzmann Machines (RBM). They rely on a hierarchical structure, that contain a visible layer, one ore more hidden layers and possibly a label layer, each with a specific number of neurons. The visible layer represents the classification input and the label layer identifies which pattern is recognised. The data processing occurs in the hidden layers. The Boltzmann Machine is restricted because there are only inter-layer but no intra-layer connections. Also the weight connections between two layers are symmetric.
In order to accomplish classification tasks, the RBMS are trained. For example the visible and label layer are clamped, so the hidden layer can adjust its connections to what it is supposed to see. [5, 4]

On Spikey we work with a hierarchical feed-forward structure, that contains one label (144 neurons), one hidden (50 neurons) and a label (6 neurons) layer. The task is to classify a subset of handwritten digits from the MNIST Dataset that are reduced to black and white 12x12 pixel images. Each visible neuron represents a pixel that is either on or off. The six label neurons stand for the six numbers 0, 1, 2, 3, 4 and 7.
Both layers are implemented in software while the hidden layer is implemented on hardware. So even though the weight connections between the layers were taken from a trained RBM (Luziwei Leng, Kirchhoff Institute for Physics, Heidelberg University) without bias, there is no feed back connec-

tion. Therefore, the used structure is not a RBM.
Since the visible neurons are clamped, we can understand them either as regular spike train input (on) or not spiking at all (off). The weight sum of all visible units to one hidden unit k can be effectively understood as a bias input. It directly translates to a spiking probability for the hidden neuron via the activation function $\sigma_k = p(z_k = 1|\mathbf{z}_{\setminus k})$ of k.

$$\begin{aligned}
\sigma_k &= \frac{1}{1+\exp(\sum_i w_i k + b_i)} && \overset{b_i=0}{\Leftrightarrow} \\
\frac{1-\sigma_k}{\sigma_k} &= \exp(\sum_i w_i k) && \Leftrightarrow \\
\ln\frac{1-\sigma_k}{\sigma_k} &= \sum_i w_i k
\end{aligned} \tag{11}$$

This also allows for one single hardware neuron to simulate all 50 hidden units. In a hardware run all spike trains are recorded and then used as input for a label layer simulations. In the end the most often spiking label neuron decides which number is classified. From these results the Classification Rate K is calculated. It quantifies the success of the task, hence the ratio of correct results against the full image set.

2 Characterisation of the Full Spikey

The following section follows closely and is compared to [7]. The setup differs in the use of both spikey halves compared to the use of one half in [7]. Also the training to testing image ratio is 100:20 instead of 20:20 per number. For more information on software and hardware setups in the experiments look into [7].

2.1 Hardware Constraints

The use of hardware brings along some issues that have to be analyzed. The analog circuit introduces delays in spike transmissions, so called synaptic delays. The theory however is based on immediate dialog between neurons. Further the refractory mechanism of each single neuron is not necessarily stable and τ_{ref} varies form spike to spike, neuron to neuron and trial to trial. Last, we deal with asymmetric activation functions because the HCS is not reached.

Synaptic Delay: Synaptic Delays are measured through inhibitory coupling of the neuron with itself while it is permanently firing ($V_{\text{thresh}} < V_{\text{rest}}$). If the transmission was immediate, we would observe an instant pull towards the reverse potential E_{inh}. The delay allows the membrane potential to drive towards its resting potential before the spike arrives. The time difference between the spike peak and the second peak is the synaptic delay, assuming τ_{ref} is very small.
In [7] the neuron to neuron average was $\tau_{\text{syn}} = 1.48 \pm 0.34$ms for the left Spikey half. A subset of 22 neurons of the full Spikey averages in $\tau_{\text{syn}} = 2.22 \pm 0.41$ms. The synaptic delay is a synapse line driver dependent issue and in the sample of both halves, two line drivers are in use, one for each half. Averaging over neurons of the first half results in $\tau_{\text{syn}} = 1.99 \pm 0.22$ms which is better. Another reasons might be the temperature difference of both tests as the first measurement was conducted in summer and the second in winter. We conclude, the results are consistent considering the experimental setup is different.

Refractory Period: In order to obtain the refractory period the function

$$V = V_{\text{reset}} + a * \Theta(t - \tau_{\text{ref}}) * (t - \tau_{\text{ref}}) \tag{12}$$

was fitted spike by spike on the membrane trace of a regularly spiking neuron. Θ is the Heaviside-Step Function.
For $I_{\text{cb}} = 0.003\mu A$ the sample subset of 50 neurons fluctuates with a relative uncertainty of 6%. This fits well to the the result $(7.56 \pm 9.5)\%$ in [7].
In general, though, the refractory period was shorter (smaller than $2ms$) compared to $\tau_{\text{ref}} = 3.8 \pm 1.3$ in [7].

Asymmetric Activation Function: The effective time constant τ_{eff} depends on the noise input rate and the membrane time constant τ_{m}. Latter is bounded by the maximal possible conductance on the chip. Another factor that leads to asymmetry is the voltage difference between reset and threshold potential. On hardware the voltage difference cannot become infinitely small due to parameter fluctuations. Therefore, the neuron needs some rising time T before it can fire again.

$$T = \tau_{\text{m}} * \ln \frac{V_{\text{reset}} - V}{V_{\text{thresh}} - V} \tag{13}$$

V is the free membrane potential, approximately V_{rest}. T summarises all factors that lead to an asymmetric activation function.

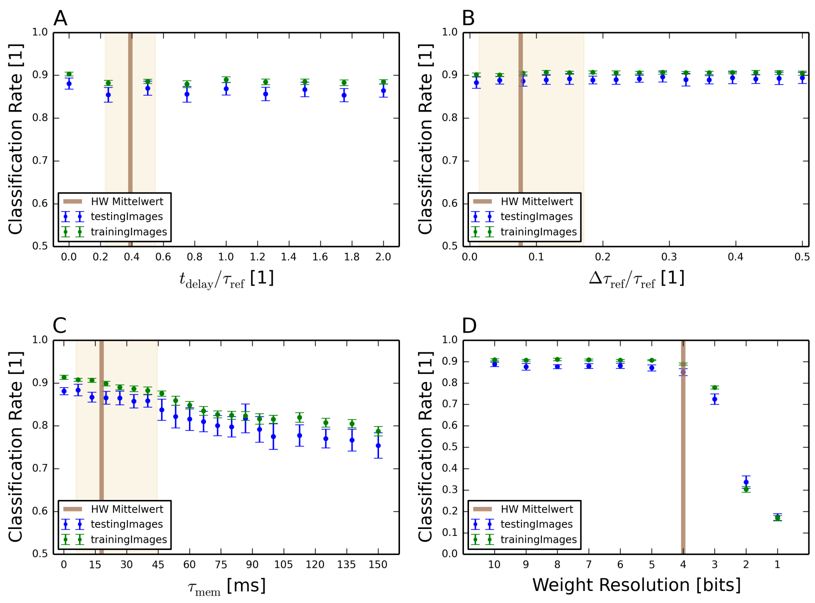

Figure 2: Influence of hardware problems to classification task: **A** Synaptic delays, **B** Unstable refractory mechanism, **C** HCS condition not fulfilled, **D** Reduced weight resolution. The brown line and shaded area mark the average Spikey value and its deviation.

T can be calculated by subtracting the refractory period from the inter-spike intervals of the previous measurement. The sample of 50 neurons averaged to $T = 1.7 \pm 1.2\ ms$ (ignoring one outlier with $23.4ms$). This fits to the result $T = 1.9 \pm 2.8\ ms$ from [7].

2.2 Robustness of the hierarchical structure

Knowing the problems we encounter on hardware, it is important to study the robustness of the hierarchical structure to them. Therefore, software simulations for each case were done (Fig. 2). Furthermore the weight resolution influence on classification was tested since only a 4bit resolution is available.

All results were averaged over 20 runs. The setup and hardware regions were taken from [7], but the training and testing images differ, hence the trained RBM has different weight connections.

Synaptic Delay: A synaptic delay range from $0.1ms$ to $2\tau_{\mathrm{ref}}$ was studied where the delays were normally distributed amongst the network neurons. The relative uncertainty was 30%.
Figure 2A shows that the structure is merely affected by synaptic delays.

Refractory Period: This simulation studies the effect of spike to spike variation of τ_{ref}. Therefore the refractory period was sampled from a gaussian distribution with uncertainties between 1% and 50% to the mean $\tau_{\text{ref}} = 15ms$. This simulation 2B also shows a certain robustness against classification failure.

Asymmetric Activation Function: To obtain asymmetric activation function, the poisson noise was reduced to $100Hz$, the difference $V_{\text{thresh}} - V_{\text{reset}}$ was set to 1V and the membrane time constant varied from $1ms$ to $150ms$. Figure 2C indicates that asymmetry does reduce the classification rate, but it still allows decent success within the Spikey region.

Weight Resolution: In this simulation the software weights from the trained RBM were binned according to the resolution. A range of 1bit to 10bit resolution was tested. Clearly, the classification rate drops for resolutions smaller than 4 bit (Fig. 2D). Therefore, the hardware should still produce reasonable results.

Spikey: In the end a simulation under full Spikey condition was started The samples from Sect. 2.1 showed that the hardware regions in the plots are still a good estimate for the full Spikey. The refractory period analysis, though, needs some further investigations to fully support this conclusion. Nevertheless, the results in the next section will show, that simulation and emulation are in good agreement.
The Classification Rate under Spikey conditions averaged over 20 runs is:

$$K_{\text{all}} = \begin{cases} 0.878 \pm 0.006 \text{ training images} \\ 0.829 \pm 0.020 \text{ testing images} \end{cases} \tag{14}$$

3 Classification on Hardware

3.1 Activation Functions

Equation (11) allows to translate the visible layer into regular spiking bias input. To set the input correctly we need the activation function of the 50 hidden neurons from the chip.
To measure the activation function, two independent runs are necessery. At first, a regular spike train and poisson noise stimulates the neuron. With increasing weight connection of the bias input, in total 16 steps, the probability for the neuron to spike rises, hence the total firing rate increases. Second the

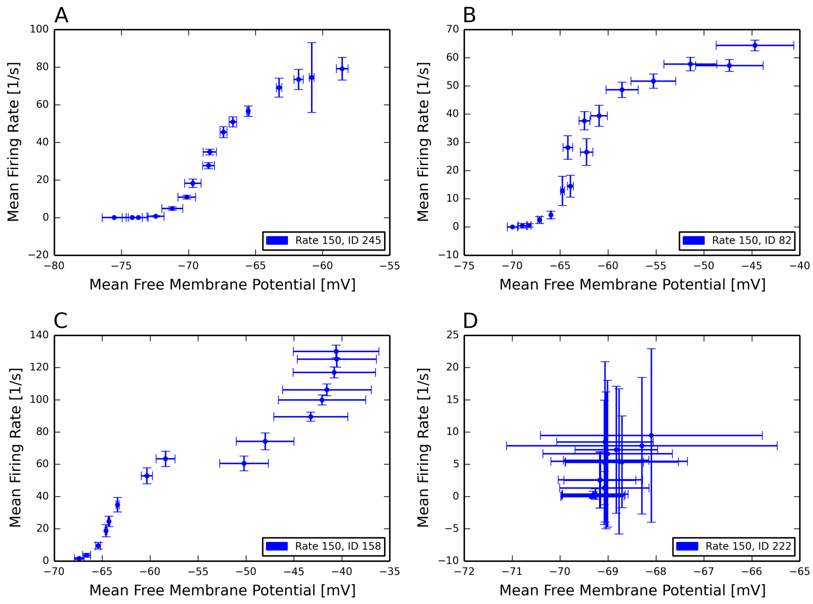

Figure 3: Measured Activation Function Categories: **A** Good, **B** Acceptable, **C** Unusable, **D** Undefinable

threshold potential is set to $V_{\text{thresh}} = 0mV$, so the neuron cannot spike at all. The free membrane potential evolves freely towards an average value, but increases with the connection strength. In the end the firing rate is plotted against the free membrane potential. Two bias rates, 150Hz and 250Hz were tested.

The activation functions of all 384 Spikey neurons were evaluated, but the results show large variations (Fig. 3). The neurons can be put into four categories. The good ones follow the sigmoid curve, even though most of them are asymmetric. The second group consists of neurons, that show the same tendency as the good ones but a few bias settings fall out of place. The last two groups make up the largest part and contain neurons that are not usable (Tab. 1).
The third group are neurons that are either dead for most weight settings, largely varying over all steps or the full sigmoid falls not in the bias step range. The fourth group shows no free membrane potential resolution, even though the firing rates often show a rising tendency with increasing weight. Interestingly, this is a sole right half phenomen (Tab. 2). This might be due to read-out problems of the membrane potential. It would be interesting to further investigate if those neurons still succeed the classification task.

The selection for the 50 hidden neurons mainly correspond to the best looking activation functions. Because of worsen neuron behaviour if the pool

	150Hz	250Hz	total
good	20	14	34
okay	59	55	114
total	79	69	148

Table 1: Number of good and okay Neurons for 150Hz and 250Hz. The total includes 32 double counts - neurons, that showed good behaviour for both rates.

150HZ	left	right	total
good	15	5	20
okay	40	19	59
type 3	137	107	246
type 4	0	61	61

250HZ	left	right	total
good	10	3	13
okay	43	12	55
type 3	139	127	265
type 4	0	50	50

Table 2: Number of good and bad Neurons divided in left and right Spikey half.

consists of too many consecutive neurons [7], a second pick argument was to prevent it by not having more than three consecutive neurons. However, this is not easily done, because the "‘good"’ neurons often followed each other, divided by a bigger pool of "‘bad"’ neurons.

One reason to use the full chip is to increase the amount of usable neurons. We would estimate to double the number by using both Spikey halves. This was not the case. Table 2 shows the comparison.

3.2 Hardware Emulation

The structure from Sect. 1.4 was implemented and evaluated. Figure 4 compares an ideal software run, the simulated hardware and the hardware runs for the new image set. The hardware runs divide into Single Neuron Runs, where one hardware neuron simulates all 50 hidden neurons, and one where the 50 hardware neurons make up the hidden layer. One neuron failed to classify and is marked red. Interestingly, this neuron (ID 61) belongs to the very promising neurons. It is unknown why the classification failed.
The results for the Single and All Neuron Run are:

$$\begin{aligned} K_{\text{single}} &= 0.83 \pm 0.10 \\ K_{\text{all}} &= 0.829 \pm 0.020 \end{aligned} \tag{15}$$

The single run is averaged over all neurons. The hardware run agrees with the expectation from the hardware simulation perfectly within significant digits.
In total the results were better than in [7] but it is not obvious if this is due

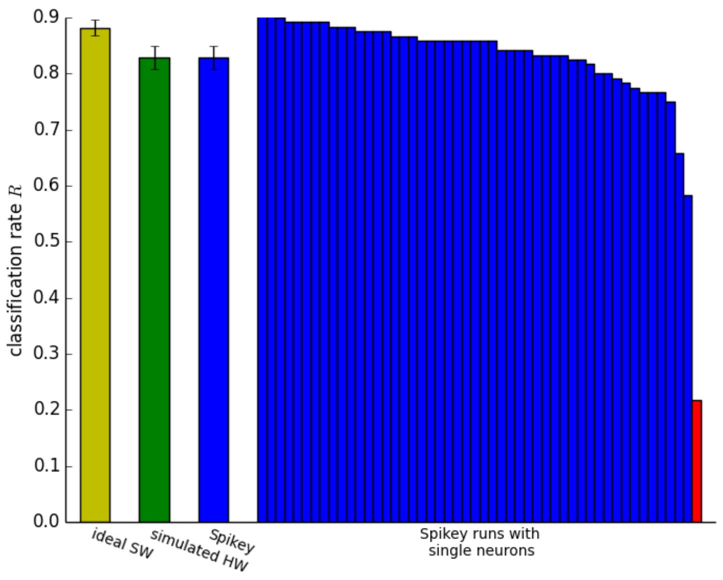

Figure 4: Comparison of Classification Runs with test image set: ideal software simulation, simulation of hardware conditions, hardware run with all neurons, hardware runs with single neurons. Except the last part the results are averaged over 20 runs.

to the new image set or the use of the full spikey.
To investigate this another hardware run was started. It classified the old image set with the corresponding trained RBM (Luziwei Leng).

$$K_{\text{oldimageSet}} = 0.782 \pm 0.008 \tag{16}$$

In this case the new neurons did not achieve better results than the neurons picked in [7]. Therefore, the classification improvement is due to the better trained RBM.

4 Discussion and Outlook

The hypothesis was that an increase of better neurons improves the classification rate. This was not the case, probably because the gain of better neurons was too small (Tab. 2). This led again to consecutive neurons that yield another issue next to largely varying activation functions (Fig. 3). However the yet untested fourth group of neurons might allow for a large pool of usable neurons. Therefore the hypothesis might still be valid.
An unlikely reason are the different hardware values measured Sect. 2.1. Apart from the fact that only a small sample was taken, the simulation of

the hardware and the actual hardware classification rate agree to good. In fact, they are identical:

$$\begin{aligned} K_{\text{sim\_hardware}} &= 0.829 \pm 0.020 \\ K_{\text{hardware\_all}} &= 0.829 \pm 0.020 \end{aligned} \tag{17}$$

The new image set shows, though, that with improvement of training, the classification on hardware improves, too. In future, the reason for the fourth group of activation function should be investigated. If the reason remains unknown, they should be at least tested with Single Neuron Runs.
So far only the hidden layer is implemented on Chip. To really use the faster hardware, all layers should be implemented on chip. Therefore, the next step is to translate the hidden-label connections into the Spikey domain and implement the label layer on chip. Next to speeding up the process, the Classification Rate could be increased with further training steps on the chip to care for hardware parameter variation.

References

[1] Lars Buesing et al. *Neural Dynamics as Sampling: A Model for Stochastic Computation in Recurrent Networks of Spiking Neurons.* English. Ed. by Olaf Sporns. SAN FRANCISCO: PUBLIC LIBRARY SCIENCE, 2011, e1002211.

[2] Mihai A. Petrovici et al. "Stochastic inference with deterministic spiking neurons". In: (Nov. 13, 2013). arXiv: `1311.3211v1 [q-bio.NC]`.

[3] Thomas Pfeil et al. "Six Networks on a Universal Neuromorphic Computing Substrate". In: *Frontiers in Neuroscience* 7 (2013), p. 11. ISSN: 1662-453X. DOI: `10.3389/fnins.2013.00011`. URL: `http://journal.frontiersin.org/article/10.3389/fnins.2013.00011`.

[4] Luziwei Leng. "Deep Learning Architectures for Neuromorphic Hardware". Masterthesis. Universität Heidelberg, 2014.

[5] Oliver Breitwieser. "Towards a Neuromorphic Implementation of Spike-Based Expectation Maximization". Masterarbeit. Universität Heidelberg, 2015.

[6] Mihai A. Petrovici. *Form vs. function. theory and models for neuronal substrates.* eng. Erstellungsdatum der Online-Ressource: 30 Jul. 2015. Heidelberg, 2016, URL: `http://nbn-resolving.de/urn:nbn:de:bsz:16-heidok-191614`.

[7] Anna Schroeder. "Struktur schafft Robustheit: Eine Untersuchung hierarchischer neuronaler Netzwerke mit unpräzisen Komponenten". Bachelorarbeit. Universität Heidelberg, 2016.

[8] Wikipedia. *Künstliches Neuronales Netz.* URL: `https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz`.