Neural Oscillations for Energy-Efficient Hardware Implementation of Sparsely Activated Deep Spiking Neural Networks

Etienne Mueller[1], Daniel Auge[1], Simon Klimaschka[1], Alois Knoll[1]

[1]Department of Informatics, Technical University of Munich
Munich, Germany
etienne.mueller@tum.de

Abstract

Spiking neural networks offer the potential for ultra-low power computations on specialized hardware, as they only consume energy when information is being transmitted. High energy efficiency is ensured when sparse temporal codes are used in which every neuron emits at most one single action potential. We present a mathematically lossless approach to convert a conventionally trained neural network to a temporal-coded spiking network. By analyzing the characteristics of spiking neurons, we eliminate the unfavorable properties for temporal coding and show the necessity of a global reference. Using neural oscillations for referencing, the resulting network shows no loss compared to the original network. Our approach makes it possible to use conventionally trained deep neural networks at the same accuracy but with the smallest necessary amount of energy.

Introduction

In recent years, deep neural networks (DNNs) have immensely increased in performance due to improved network architectures and more powerful training hardware. Because of their computational advantages, they find their way into an increasing number of applications. However, implementation in power restricted environments, such as data processing in edge devices or decision-making in autonomous driving, is limited by the available energy. Therefore, reducing the energy consumption of neural networks directly results in longer battery life or an increased range in these cases.

To reduce energy consumption, today's main approaches include the development of new network architectures and the optimization of the hardware running them. Another approach, that has gained traction in recent years, is the use of biologically inspired spiking neural networks (SNNs), which additionally also show higher computational power (Maass 1997). By communicating with short all-or-nothing pulses instead of continuous-valued activation functions, these networks also promise low-powered hardware implementation. The silicon implementation of SNNs, neuromorphic hardware, in theory only consumes energy when information is being transmitted. Multiple chips, e.g. Intel Loihi (Davies et al. 2018), IBM TrueNorth (Merolla et al. 2011) and SpiNNaker (Painkras et al. 2013), are available for research.

Despite these advantages, SNNs commonly show inferior performance compared to artificial neural networks (ANNs) and are therefore not widely used today. Because of their temporal dynamics and the resulting non-differentiable activation functions, using powerful gradient descent-based training algorithms is not directly possible. As a result, the three main training approaches are as follows: (1) unsupervised learning approaches such as spike-timing-dependent plasticity (Markram et al. 1997; Caporale and Dan 2008), (2) supervised learning approaches that try to adapt gradient descent based backpropagation (Pfister et al. 2006; Neftci et al. 2014; Neftci, Mostafa, and Zenke 2019), and (3) conversion approaches that train ANNs with backpropagation and map the weights to a spiking network. Among these three approaches, conversion has shown the best performance on popular benchmarks (Tavanaei et al. 2019).

While early work demonstrated the difficulties of mapping sigmoid-activated neural networks to SNNs (Perez-Carrasco et al. 2013), the use of rectified linear units (ReLU) (Nair and Hinton 2010) resulted in a near lossless conversion (Cao, Chen, and Khosla 2014). This is caused by the property that both the ReLU activation function and the firing rate of spiking neurons increase linearly with their input. Additional approaches for weight normalization (Diehl et al. 2015) and operators for convolutional neural networks (CNNs) (Rueckauer et al. 2017) enabled the use of deep SNNs, including spiking variants of high performing object detection DNNs, e.g. YOLO (Kim et al. 2019), ResNet (Hu et al. 2018), and RetinaNet (Mueller et al. 2021).

However, achieving comparable performance to the original ANN comes with a side effect: as the information in the converted spiking networks is encoded in the firing rates, highly active neurons fire multiple times during inference, resulting in high energy consumption. Whereas early work estimated reduced energy consumption of $10^1 - 10^5$ times from digital computers to silicon neurons (Poon and Zhou 2011), recent analyses concluded that only very low spiking activity justifies the use of spiking networks (Davidson and Furber 2021). A practical approach measured the energy consumption of a very small ANN deployed on an accelerator chip and compared it to the converted network running on Intel Loihi, reporting a $4.11\times$ smaller energy consump-

tion (Blouw and Eliasmith 2020).

To achieve the necessary level of sparsity, research intensifies the use of temporal coding techniques. By encoding the information in the period leading up to the spike, neurons with positive activations only need to fire once for computation. While learning algorithms show large potential (Chen et al. 2021; Zhou et al. 2021), conversion approaches contend with a loss of information of low activated neurons and need additional adjustments, resulting in a larger amount of spikes (Rueckauer and Liu 2018).

In this paper, we present a method that directly maps ReLU activation to temporal coded spiking networks. We address the loss of information by temporarily separating input and spike generation. Our contributions are as follows:

- Analyzing the properties of spiking neurons and comparing them to linear activation functions. Our analysis shows that converting ANNs to temporal coded SNNs requires the use of global referenced non-leaky integrate-and-fire neurons with input spike adaption.
- Proposing a network model based on previous findings, using neural oscillation as a global reference for spike generation. Two methods for the input spike adaption are presented: (1) pulse duration adaption and (2) amplitude adaption.
- Evaluating our approach and comparing it to previous methods in terms of spiking operations and scalability.

This paper is organized as follows: Section 2 shows related work for the conversion of ANNs to temporal coded spiking networks. In Section 3, we analyze the information coding in spiking neurons and show how to use them for temporal conversion. Then, Section 4 presents two methods to combine the approaches in SNNs and are evaluated in Section 5. We conclude the paper with a discussion in Section 6.

Related Work

In previous work on the conversion of ReLU-activated ANNs to temporal coded spiking networks, Rueckauer et al. (Rueckauer and Liu 2018) proposed the *TTFS base* method, which modeled the incoming spikes as a weighted permanent input current. When the resulting membrane potential crosses the threshold, a spike is emitted and the neuron is reset to zero. To prevent additional firings the refractory period is made very long. The main disadvantage of this method is that neurons can potentially fire their spikes prematurely, e.g. when a positive input raises the membrane potential over the threshold before a negative input arrives, which would balance out the first input out. To remedy this situation, Rueckauer et al. proposed *TTFS dyn thres*, where the neurons feature a dynamic threshold that is increased by the amount of missing input. To signal the missing input, neurons fire twice: the first time as a signal and the second time containing the temporal information.

In their experiment on the MNIST dataset with a 5 layer LeNet5-CNN, Rueckauer et al. reported roughly 1% of loss for *TTFS base* and 0.8% for *TTFS dyn thres*. A third approach, named *TTFS clamped*, relied on re-training the entire network with a modified ReLU activation function, that clips the lower activation and subsequently performs better.

Temporal Coding in Spiking Neurons

For a lossless conversion of ReLU activated ANNs to temporal codes, no spike can be emitted before all inputs have arrived. Therefore, we propose a method where each layer has two time windows with the same duration for: (1) receiving incoming spikes and (2) emitting outgoing spikes (if any). As a result, the necessity of a "driving force" arises that ensures a spiking of the neuron as long as its membrane potential is positive.

Biological Inspiration

Biological neurons dispose of multiple ion channels that open when a positive input current is present, leading to an ion flow that increases the neuron's membrane potential. When the input stops, due to leak channels the membrane potential returns to its resting state. When the membrane potential reaches a threshold, an avalanching effect creates an action potential that is forwarded to subsequent neurons. Accordingly, the membrane potential resets and can not spike during a short refractory period.

The relation between input current and membrane potential has long been explored and described as a function of four coupled differential equations in the Hodgkin-Huxley model (Hodgkin and Huxley 1952). Because the shape of the spike is always similar, the assumption that the action potential itself does not contain any information. Multiple variants have been presented that can model different properties of biological neurons and reduce computational complexity (Izhikevich 2004). The simplest being the leaky-integrate-and-fire (LIF) neuron, with the membrane potential being calculated as follows $u(t)$ (Gerstner et al. 2014):

$$u(t) = u_{rest} + RI_0 \left[1 - exp\left(-\frac{t}{\tau_m}\right)\right] \tag{1}$$

It can be approximated as a capacitor C in parallel with a resistance R driven by the current $I(t)$ with the time constant $\tau_m = RC$.

Conversion of Input Signals

To utilize common datasets with spiking networks, the input first has to be converted to spikes. One approach in rate-coded networks is to use Poisson generators that generate spikes with the probability of the input value. In the case of temporal coding this is not possible, as only the first spike is from interest and there is no averaging over time. Another approach is to feed the input values as constant current to a non-leaking neuron. This leads to a linear increase of the membrane potential and a subsequent spike generation. By normalizing the input the spike will occur either directly at the beginning or at the end of a specific time window. For simplicity, we assume an even distribution of the spikes during the time window, so it is directly comparable to the input data.

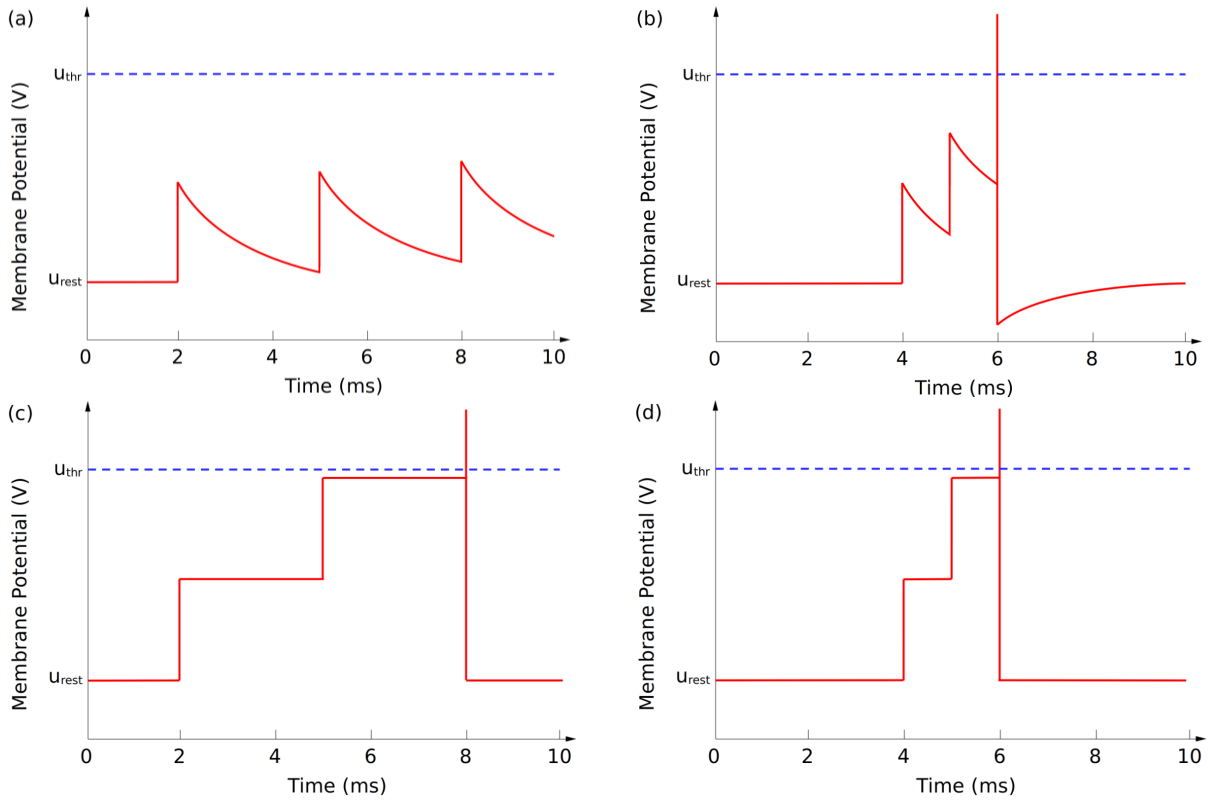

Figure 1: Membrane potential of LIF neurons (with refractory period after action potential, a and b) and non-LIF neurons (reset-to-zero after action potential, c and d) as a response to input spikes at [2, 5, 8] ms (a and c) and [4, 5, 6] ms (b and d).

Relation between ReLU and Spiking Neurons

Biologically inspired models innately react to different patterns in data compared to ANNs. For example, the two input patterns [0.8, 0.5, 0.2] and [0.6, 0.5, 0.4] will both result in an activation of $a = 1.5$ for a ReLU activated neuron. Conversely, when representing the input as spikes spaced in a 10 ms time window, a LIF neuron will remain silent for the first case (fig. 1a) but generate an action potential in the latter one (fig. 1b). This coding scheme, often referred to as *interspike interval* coding, is a direct result of the neuron's leak. Furthermore, as the leak is exponentially decaying, an input's impact on the membrane potential at spike generation will be dependent on the membrane potential and thus can not be used for the conversion, as in ANNs the input to the neurons has a fixed value independently of the other inputs.

When using integrate-and-fire (IF) neurons, it can resolve the unwanted properties of the leakage and show a better response to the previously discussed input, but still shows issues that are not beneficial for temporal conversion:

- *No temporal information:* Removing the leakage also removes the temporal information of the input. As can be seen in fig. 1c and 1d, the membrane potential has a constant value after each input, and the input times can be arbitrarily moved before the third spike and show the same output time.
- *Multiple pattern recognition:* Many different input patterns can result in generating a spike at the same time, e.g. a few large spikes or many small ones. With that, one single neuron can be used to detect multiple patterns even on different hierarchical levels. Neurons in ANNs are trained to recognize a very specific pattern and only one linearly independent input can maximize the resulting activation.
- *Discarding low activations:* When a spiking neuron does not receive sufficient input, the membrane potential will not cross the threshold and thus remains silent. For example, if the first or second input of fig. 1b was zero, the threshold would not be reached. In contrast, ReLU will have a certain value, as long as the total input is positive.
- *Premature spiking:* when an action potential is generated, the information of later input is lost (Rueckauer and Liu 2018). For example, a large negative input at 9ms in fig. 1c or 1d will have no impact on the spike although the total inputs should not have created any response.

Approach

To compensate for the issues of IF neurons, it raises the need for certain requirements to adapt for temporal conversion:

Fixed Windows To prevent the premature spiking of neurons, all input must be received before emitting a spike. This can be achieved by using a *listening phase*, where the input to the neuron affects its membrane potential but does not

generate an action potential. In a subsequent *transmitting phase* with the same duration, an action potential is generated with the exact time being depending on the extent of the membrane potential.

Normalization To ensure that no spikes are generated in the first window, the weights of the original ANN get normalized. Therefore, if a neuron receives the maximum possible input in the first time window, its membrane potential will be just below the spiking threshold. In the second window, the membrane potential will be linearly increased. Neurons with high potential will then emit a spike early in that time frame, while neurons with a low potential will fire late in that window. Neurons with negative potential will only fire after that time frame is over and thus not have any effect on the subsequent neurons.

Neural Oscillation Since no input is present in the transmitting phase, the membrane potential remains at a constant value. To push a positive membrane potential over the threshold a linear increase is needed. Accordingly, we implement a global referenced neural oscillation in the form of a continuous input current that switches between positive and negative after each phase. It is set such that at the end of the transmitting phase, if no input is present the membrane potential would only reach the threshold without emitting a spike. A positive membrane potential after the listening phase then results in crossing the threshold, with higher potentials spiking earlier and lower potentials later.

Weighting of Input As no temporal information is present when using IF neurons, the incoming spikes have to be adapted depending on when they occur during the listening phase. As the amplitude of the voltage response in spiking neurons depends on the total charge of the spike $q = \int I(t)dt$ (Gerstner et al. 2014), this gives two options to adjust: (1) the amplitude of the incoming spike or (2) the duration of the pulse. Both can be accomplished by coupling them to the neural oscillation.

Bias In rate-coded conversion, the bias is commonly treated as a constant current to the neuron. When using neural oscillations, that would lead to a spike during the first cycle when a positive bias is present. To circumvent this issue we use an additional neuron in each layer that is fully activated and send a spike weighted by the value of the bias to all neurons in the subsequent layer. See fig. 2

Refractory Period To prevent additional spiking in later oscillating cycles, the refractory period of the neurons are set to a value higher than the total runtime of the network.

Methods

Adaption can be done in two ways: (1) lowering the amplitude of the spike or (2) adapt the duration of the spike

$$a_i^l := max(0, \sum_{j=1}^{l-1} W_{ij}^l a_j^{l-1} + b_i^l) \tag{2}$$

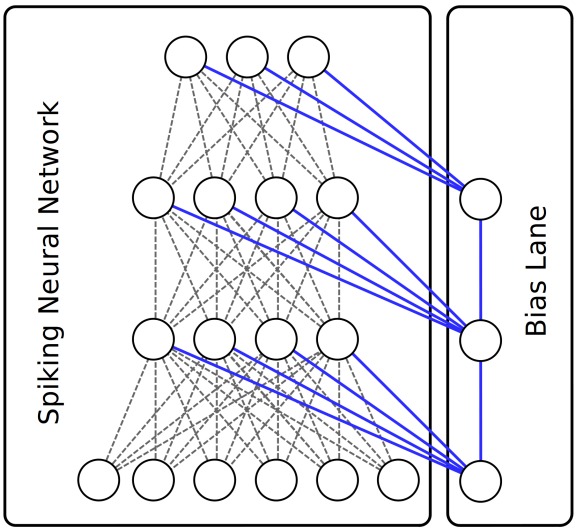

Figure 2: Bias Lane: one additional neuron per layer that injects an additional pulse weighted by the value of the bias.

Spike Amplitude Adaption.

The later a spike arrives, the lower its amplitude has to be. By using e.g. an adaptive resistor that is coupled to the neural oscillation, it can reduce the spike amplitude the later it occurs during the listening phase. Because the spikes result in steps of the membrane potential, the listening phase has to start at $-v_{thr}$, reach 0 between phases, and grow to u_{thr} until the end of the second phase. The subsequent layers require a phase shift. Membrane potential $u_i(t)$ of neuron i:

$$u_i(t) = \sum W_{ij} \frac{t_j}{T} \delta_j + b_i^l \tag{3}$$

with T being the duration of the listening phase and δ the Dirac delta function.

Due to the linear increase of the membrane potential during the transmission phase, the resulting spike time relative to its time frame can then be calculated with:

$$t_i(u) = (u_{thr} - u_{l+1})T \tag{4}$$

with u_{l+1} being the membrane potential at the beginning of the transmission phase of layer l and the beginning of the listening phase of layer $l+1$, respectively.

Pulse Duration Adaption

By adapting the duration of the pulses, after an input occurred a constant current is fed to the neuron during the remainder of the listening phase. Therefore, early spikes have a larger impact on the membrane potential at the end of the phase than later ones. This can be done by making the pulse duration as long as the duration of one oscillation. Negative current from oscillation during the listening phase, positive during transmitting. As the positive and negative phases are reversed for the connecting layer, the sign of the pulses will also be reversed and the threshold set to $-u_{thr}$. The membrane potential $u_i(t)$ of the neuron i at the end of the listening phase then can be calculated with:

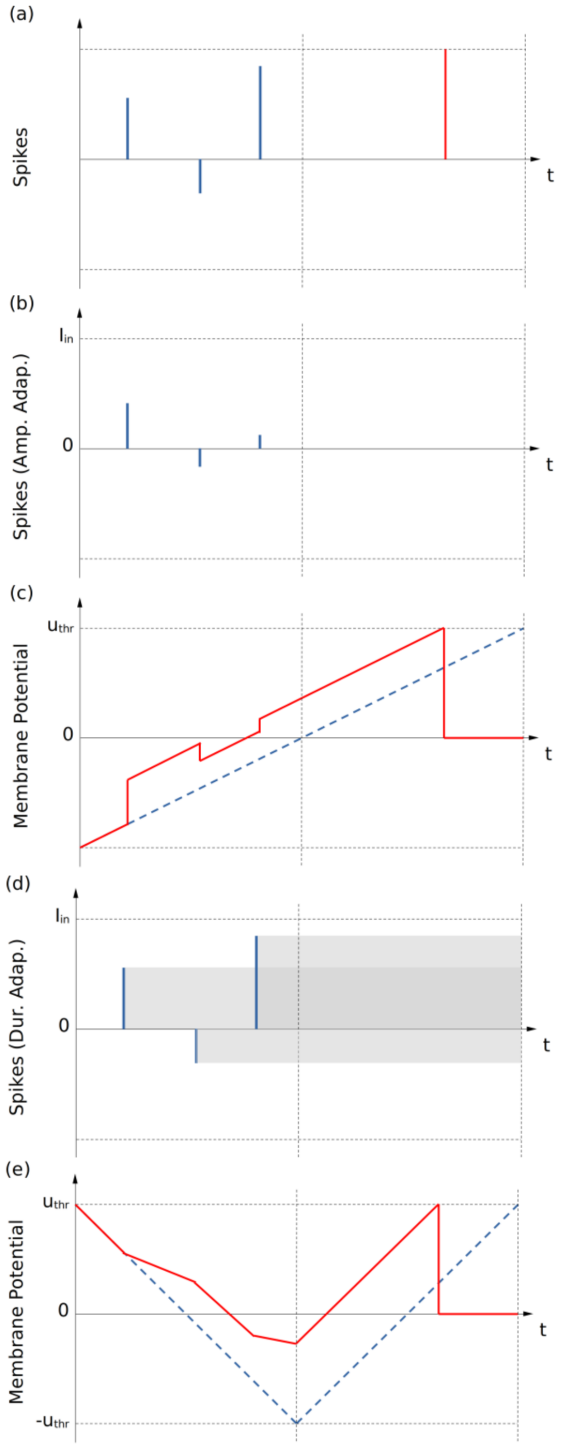

Figure 3: Neural oscillation method (left half: listening phase, right half: transmission phase). (a) Input spikes during listening (left) and action potential generated from the two methods (right). (b) Input and (c) resulting membrane potential with spike adaption method. (d) Input and (e) resulting membrane potential with pulse duration adaption.

$$u_i(t) = \sum W_{ij}(T - \Delta t_j)q_p + b_i^l q_p \tag{5}$$

Δt_j the time of the incoming pulse relative to the start of the phase, q_p the total charge of a pulse. The spike time is calculated like in eq. 4.

Experiments

As the approach is mathematically the same as traditional machine learning, comparing the accuracy in a simulation is not meaningful, as it directly depends on the accuracy of the original ANN. Instead, we compare the inference time to previous approaches and link the spiking operations to previous work that measures that of rate-coded conversion methods.

Comparing to Temporal Coded Conversion

Previous work (Rueckauer and Liu 2018) did not rely on global reference and thus showed problems with premature spiking of neurons. An adapted version could reduce the problem by using dynamic thresholds, which leads to doubling the number of spikes. We train two neural networks with different depths on the MNIST dataset (LeCun et al. 1998) and compare this approach to our method.

Model	Network	Loss	# spikes
TTFS base	LeNet-5	3.85%	2011
TTFS dyn thresh	LeNet-5	1.77%	3867
This work	LeNet-5	-	2854
TTFS base	9-layer CNN	9.18%	4670
TTFS dyn thresh	9-layer CNN	6.36%	9132
This work	9-layer CNN	-	8234

Table 1: Comparison of our method to previous temporal coding approaches (Rueckauer and Liu 2018) on the MNIST dataset. The LeNet-5 Architecture contains 7620 neurons (7625 including the bias lane) and the 9-layer CNN (a variant of the LeNet-5 architecture with additional convolutional layer) in a total 13900 neurons (13909 including the bias lane).

In recreating the results of the previous work, we achieved similar performance as reported for the LeNet-5 network architecture (see tab. 1). Our approach shows the same accuracy of 98.7% as the original ANN, while the TTFS base and TTFS dyn thresh methods showed a loss of 3.85% and 1.77%, respectively. The number of spikes roughly doubles for TTFS dyn thresh and our approach is roughly in the middle.

When scaling the network to a deeper architecture, namely a 9-layer CNN based on LeNet-5 with additional convolutional layers, the TTFS base and TTFS dyn thresh methods show large loss. As pointed out by Rueckauer et al., the TTFS base method suffers from prematurely spiking.

For the TTFS dyn thresh method, we assume that although the use of an additional spike per neuron, it can reduce the problem but not completely remove it, as the first spike is only fired when the first input arrived. When a neuron in a subsequent layer has already fired its second spike, it will also lose the information. By scaling the network, the interval between the spike times of different layers increases and reinforces this problem. In essence, both approaches cut off lower activated input the deeper the network gets.

For the deep network, the TTFS base method requires the least amount of spikes and the mose TTFS dyn thresh. Our approach is again in between, but closer to the latter.

Comparison to Rate Coded Conversion

For comparison with a rate-coded conversion, we train a neural network on a portion of the speech commands dataset (Warden 2018), with a similar network architecture as in (Blouw and Eliasmith 2020) and with an input layer of 3920 neurons and two hidden layers with 256 neurons. As in the original work a Hybrid SNN was used where the input was digitally processed and after the first layers spikes were generated, we only count the spikes from that layer as well.

Model	ANN	SNN	# spikes
(Blouw et al. 2020)	81.8%	81.0%	61362
This work	82.1%	82.1%	4081

Table 2: Comparison of our method to rate coded approaches on the speech recognition dataset. The architecture contains an input layer with 3920 neurons and two hidden layers with 256 neurons. Our approach needs one additional neuron per layer.

While our ANN model achieves comparable 82.1% in accuracy compared to 81.8% in the original work, our approach stays at the same accuracy after conversion, whereas the rate coded approach drops to 81.0%. Whereas the original work needed 61362 SOPs, our approach reduces it by a factor of roughly 15 to only 4081 SOPs averaged over the test set. It is to mention, that nearly all the neurons in the input layer generated a spike and less than 200 spikes were generated in the hidden and classification layer.

The original work compared the energy consumption of running the network as an ANN on a neural accelerator and as a converted SNN on Intel Loihi. The SNN reported a $4.11\times$ energy reduction compared to the original network. The $15\times$ fewer spikes in our approach should be considered with caution, as the additional need for a global reference requires additional power and thus can not directly be used as a factor for possible energy reduction.

Discussion

In this paper, we propose a method for the conversion of continuous activated neural networks to temporal coded SNNs. By using globally referenced neural oscillation it enables to map the exact value of the ReLU activation function into a listening phase and then emit a spike during a transmission phase. As this approach is mathematically equivalent to the calculation of ANNs, the conversion is in fact lossless.

We validate the performance of the method in two experiments and show that it scales to deep network architectures and also reduces the number of spikes compared to rate-coded conversion by a factor of 15. Although the performance is lossless in theory and in simulation, deploying the method on neuromorphic hardware can lead to accuracy loss. When running on digital neuromorphic hardware, the network is quantized and the accuracy is subsequently dependent on the time steps of the simulation. On analog neuromorphic hardware, noise can also infer with the accuracy and also would need to run the network slower. The use of a global reference is necessary for this conversion approach, but might be incompatible with current available neuromorphic hardware. Furthermore, the use of neural oscillations results in additional energy consumption.

Overall, our method shows a lossless approach for conversion to temporal coded neural networks and enables new approaches in neuromorphic hardware.

Acknowledgment

The authors would like to thank Infineon Technologies AG for supporting this research.

References

Blouw, P.; and Eliasmith, C. 2020. Event-Driven Signal Processing with Neuromorphic Computing Systems. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8534–8538. Barcelona, Spain: IEEE. ISBN 978-1-5090-6631-5.

Cao, Y.; Chen, Y.; and Khosla, D. 2014. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *International Journal of Computer Vision*, 113(1): 54–66.

Caporale, N.; and Dan, Y. 2008. Spike Timing-Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, 31(1): 25–46.

Chen, Y.; Qu, H.; Zhang, M.; and Wang, Y. 2021. Deep Spiking Neural Network with Neural Oscillation and Spike-Phase Information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7073–7080.

Davidson, S.; and Furber, S. B. 2021. Comparison of Artificial and Spiking Neural Networks on Digital Hardware. *Frontiers in Neuroscience*, 15.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; Liao, Y.; Lin, C.-K.; Lines, A.; Liu, R.; Mathaikutty, D.; McCoy, S.; Paul, A.; Tse, J.; Venkataramanan, G.; Weng, Y.-H.; Wild, A.; Yang, Y.; and Wang, H. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 38(1): 82–99.

Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.; and Pfeiffer, M. 2015. Fast-Classifying, High-Accuracy Spiking Deep Networks through Weight and Threshold Balancing.

In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Gerstner, W.; Kistler, W. M.; Naud, R.; and Paninski, L. 2014. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press.

Hodgkin, A. L.; and Huxley, A. F. 1952. A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve.

Hu, Y.; Tang, H.; Wang, Y.; and Pan, G. 2018. Spiking Deep Residual Network. *arXiv:1805.01352 [cs]*.

Izhikevich, E. 2004. Which Model to Use for Cortical Spiking Neurons? *IEEE Transactions on Neural Networks*, 15(5): 1063–1070.

Kim, S.; Park, S.; Na, B.; and Yoon, S. 2019. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07): 11270–11277.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Maass, W. 1997. Networks of Spiking Neurons: The Third Generation of Neural Network Models. *Neural Networks*, 10(9): 1659–1671.

Markram, H.; Lübke, J.; Frotscher, M.; and Sakmann, B. 1997. Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science*, 275(5297): 213–215.

Merolla, P.; Arthur, J.; Akopyan, F.; Imam, N.; Manohar, R.; and Modha, D. S. 2011. A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 1–4. San Jose, CA, USA: IEEE. ISBN 978-1-4577-0222-8.

Mueller, E.; Hansjakob, J.; Auge, D.; and Knoll, A. 2021. Minimizing Inference Time: Optimization Methods for Converted Deep Spiking Neural Networks. In *International Joint Conference on Neural Network*, 8.

Nair, V.; and Hinton, G. E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, 8.

Neftci, E.; Das, S.; Pedroni, B.; Kreutz-Delgado, K.; and Cauwenberghs, G. 2014. Event-Driven Contrastive Divergence for Spiking Neuromorphic Systems. *Frontiers in Neuroscience*, 7.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. Surrogate Gradient Learning in Spiking Neural Networks. *arXiv:1901.09948 [cs, q-bio]*.

Painkras, E.; Plana, L. A.; Garside, J.; Temple, S.; Galluppi, F.; Patterson, C.; Lester, D. R.; Brown, A. D.; and Furber, S. B. 2013. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation. *IEEE Journal of Solid-State Circuits*, 48(8): 1943–1953.

Perez-Carrasco, J. A.; Zhao, B.; Serrano, C.; Acha, B.; Serrano-Gotarredona, T.; Chen, S.; and Linares-Barranco, B. 2013. Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing-Application to Feedforward ConvNets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11): 2706–2719.

Pfister, J.-P.; Toyoizumi, T.; Barber, D.; and Gerstner, W. 2006. Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning. *Neural Computation*, 18(6): 1318–1348.

Poon, C.-S.; and Zhou, K. 2011. Neuromorphic Silicon Neurons and Large-Scale Neural Networks: Challenges and Opportunities. *Frontiers in Neuroscience*, 5.

Rueckauer, B.; and Liu, S.-C. 2018. Conversion of Analog to Spiking Neural Networks Using Sparse Temporal Coding. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Frontiers in Neuroscience*, 11.

Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S. R.; Masquelier, T.; and Maida, A. 2019. Deep Learning in Spiking Neural Networks. *Neural Networks*, 111: 47–63.

Warden, P. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv:1804.03209 [cs]*.

Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-Coded Deep Spiking Neural Network with Easy Training and Robust Performance. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11143–11151.