Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes

Christoph Stöckl[1], Wolfgang Maass[1,*]

July 10, 2020

[1]*Institute of Theoretical Computer Science, Graz University of Technology, Inffeldgasse 16b, Graz, Austria*
[*] *To whom correspondence should be addressed; E-mail: maass@igi.tugraz.at.*

Abstract

Spike-based neuromorphic hardware is a promising option for reducing the energy consumption of image classification, or more generally of inference in large neural networks that have been trained by deep learning. A drastic reduction of this energy consumption is especially needed for implementing state-of-the-art results of deep learning in edge devices. However direct training of deep feedforward spiking neural networks is difficult, and previous methods for converting trained artificial neural networks to spiking neurons required too many spikes. We show that a substantially more efficient conversion from artificial neural networks to spike-based networks is possible if one optimizes the spiking neuron model for that purpose, and enables it to use the timing of spikes to encode information. This method allows us to significantly advance the accuracy that can be achieved for image classification with spiking neurons, and the resulting networks need on average just two spikes per neuron for classifying an image. In addition our new conversion method drastically improves latency and throughput of the resulting spiking networks.

Introduction

Spiking neural networks (SNNs) are currently explored as possible solution for a major impediment of more widespread uses of modern AI in edge devices: The energy consumption of the large state-of-the-art artificial neural networks (ANNs) that are produced by Deep Learning. This holds in particular for the Convolutional Neural Networks (CNNs) that are commonly used for image classification, but also other application domains. These ANNs have to be large for achieving top performance, since they need to have a sufficiently large number of parameters in order to absorb enough information from the huge data sets

on which they have been trained, such as the 1.2 million images of the ImageNet2012 dataset. Inference with standard hardware implementations of these large ANNs is inherently power-hungry [García-Martín et al., 2019]. Spiking neurons have been in the focus of the development of novel computing hardware for AI with a drastically reduced energy budget, partially because the giant SNN of the brain –consisting of about 100 billion neurons– consumes just 20W [Ling, 2001]. Spiking neurons output trains of stereotypical pulses that are called spikes. Hence their output is very different from the analog numbers that an ANN neuron produces as output. Most spiking neuron models that are considered for implementation in neuromorphic hardware are inspired by simple models for spiking neurons in the brain.

But whereas large ANNs, trained with ever more sophisticated Deep Learning algorithms on giant data sets, approach –and sometimes exceed– human performance in several categories of intelligence, the performance of SNNs is lagging behind. There is some hope that this gap can be closed for the case of recurrent neural networks, since BPTT and e-prop applied to recurrent SNNs with adapting neurons appears to capture most of the performance recurrent ANNs with sparsely active spiking neurons [Bellec et al., 2019]. But the problem to produce SNNs that achieve similar performance as ANNs with few spikes persists for feedforward networks. Feedforward convolutional neural networks (CNNs) that achieve really good image classification accuracy tend to be very deep and very large, and training corresponding deep and large feedforward SNNs has not been able to reach similar classification accuracy. Problems with the timing of spikes and precision of firing rates on higher levels of the resulting SNNs have been cited as possible reasons. One attractive alternative is to simply take a well-performing trained CNN and convert it into an SNN –using the same connections and weights. The most common –and so far best performing— conversion method was based on the idea of (firing-) rate coding, where the analog output of an ANN unit is emulated by the firing rate of a spiking neuron [Rueckauer et al., 2017]. This method had produced so far the best SNN results for image classification. But the transmission of an analog value through a firing rate tends to require a fairly large number of time spikes, which reduces both latency and throughput of the network. Furthermore, the resulting SNN tends to produce so many spikes that its energy-advantage over non-spiking hardware gets lost. Finally, a rate-based ANN-to-SNN conversion can not be applied to those ANNs that currently achieve the highest accuracy on ImageNet, EfficientNets [Tan and Le, 2019], because these employ an activation function that assumes both positive and negative values: the Swish function.

We introduce a new ANN-to-SNN conversion that we call FS-conversion because it requires a spiking neuron to emit just a few spikes (FS = Few Spikes). This method is completely different from rate-based conversions, and exploits the option of temporal coding with spikes, i.e., the timing of a spike transmits extra information. Most previous proposed forms of temporal coding, see e.g. [Maass and Natschläger, 1998], [Thorpe et al., 2001], [Rueckauer et al., 2017], [Kheradpisheh and Masquelier, 2019], have turned out to be difficult to implement efficiently in neuromorphic hardware because fire time-differences between spikes need to be transmitted to downstream neurons. In contrast, an FS-conversion can be implemented with -in the worst case- log N different spike times

and spikes for transmitting integers between 1 and N. Practically, the required number of spikes can be made even lower because not all N values occur equally often. However FS-conversion requires a modified spiking neuron model, the FS-neuron, with a periodically changing firing threshold. We propose to use this optimized spiking neuron model as guidance for the next generation of neuromorphic hardware.

We will describe in the first subsection of Results the design of an FS-neuron that can emulate an ANN neuron with practically any activation function. We then demonstrate the performance of SNNs that result from FS-conversion of CNNs, on two state-of-the-art datasets for image classification: ImageNet2012 and CIFAR10.

1 Results

1.1 FS-neuron model

The FS-conversion from ANNs to SNNs requires a variation of the standard spiking neuron model, to which we refer as FS-neuron. We emulate the activation function $f(x)$ of an artificial neuron (see Fig. 1A) by a spiking neuron with a periodically changing firing threshold $T(t)$ as shown in Fig. 1B. The input-output behavior of an FS-neuron is defined by the values of parameters $T(t), h(t), d(t)$ for $t = 1, ..., K$. These are optimized to emulate the activation function $f(x)$ of the given ANN neuron, see Fig. 2 for the case of the Swish function of EfficientNet, and Fig. 3 for the case of a standard sigmoid function. To emit a spike at time t, a neuron's membrane potential $v(t)$ has to surpass the current value $T(t)$ of its firing threshold. We assume that the membrane potential $v(t)$ has no leak, but is reset to $v(t) - h(t)$ after a spike at time t. We denote the spike train that this neuron produces by $z(t)$, i.e., $z(t) = 1$ if the neuron fires at step t, else $z(t) = 0$. Each time step of a non-spiking artificial neuron in the given feedforward ANN is simulated by K time steps $1, ..., K$ of this FS-neuron. Expressed in formulas, the membrane potential $v(t)$ evolves during these K steps according to

$$v(t+1) = v(t) - h(t)z(t). \tag{1}$$

The spike output $z(t)$ of an FS-neuron for gate input x can be defined compactly by

$$z(t) = \Theta(v(t) - T(t)) = \Theta\left(\left(x - \sum_{j=1}^{t-1} h(j)z(j)\right) - T(t)\right), \quad t = 1, ..., K, \tag{2}$$

where Θ denotes the Heaviside step function. The total output $\hat{f}(x)$ of the FS-neuron during these K time steps can be written as:

$$\hat{f}(x) = \sum_{t=1}^{K} d(t)z(t). \tag{3}$$

An illustration of the model can be found in Figure 1B. The computing strategy of an FS-neuron is defined by its parameters $T(t), h(t), d(t)$ for $t = 1, ..., K$. For emulating the

ReLU activation function one can choose these values so that they define a coarse-to-fine processing strategy for all input values x that lie below some upper bound, as described in section 1.2.2. For emulating the Swish function of EfficientNet one achieves a better FS-conversion if the parameters are chosen in such a way that they enable iterative –and thereby more precise– processing for the range of inputs between -2 and 2 that occur most often as gate inputs x in EfficientNet.

All FS-neurons that emulate ANN neurons with the same activation function can use the same parameters $T(t)$, $h(t)$, $d(t)$, while the factor w in the weights of their output spikes is simply lifted from the corresponding synaptic connection in the trained ANN. Note that the number of neurons and connections in the network is not increased through the FS-conversion.

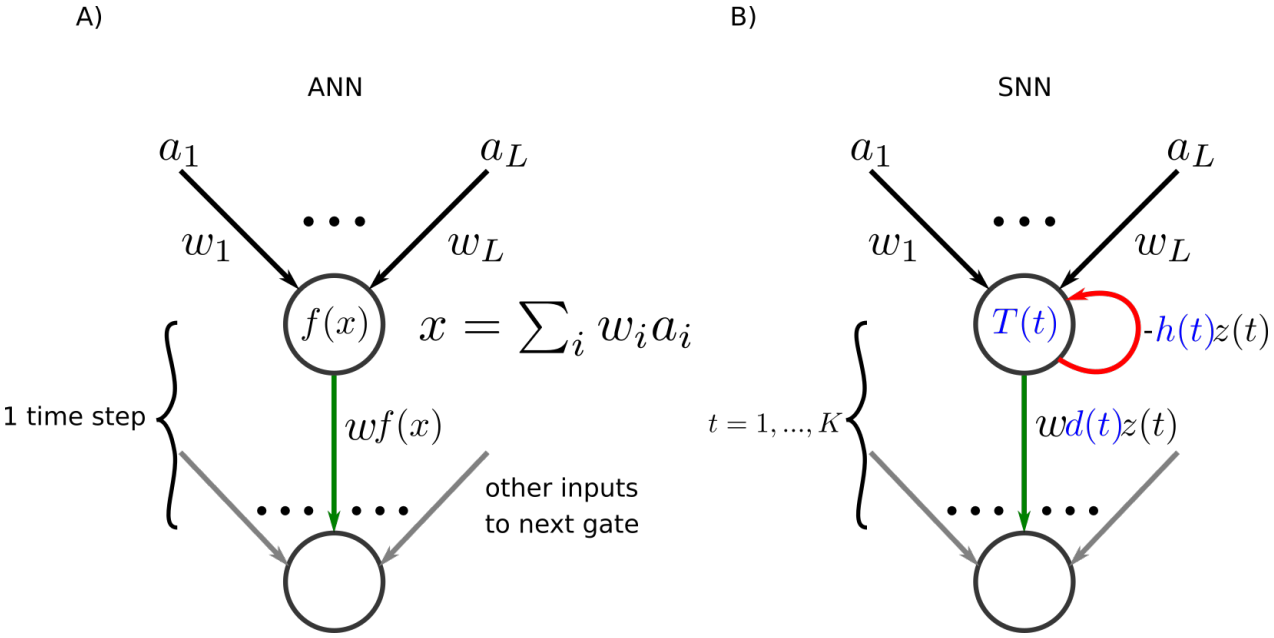

Figure 1: Conversion of an ANN neuron into an FS-neuron.
A) *A generic ANN neuron with activation function $f(x)$ that is to be emulated.*
B) *An FS-neuron which emulates this ANN neuron in K time steps $t = 1, ..., K$. Its output spike train is denoted by $z(t)$.*

Both the TensorFlow code and the parameters of the FS-neurons are available online*.

1.2 Application of the FS-conversion to the classification of images from ImageNet

The ImageNet data set [Russakovsky et al., 2015] has become the most popular benchmark for state-of-the-art image classification in machine learning (we are using here the

*https://github.com/christophstoeckl/FS-neurons

ImageNet2012 version). This data set consists of 1.281.167 training images and 50.000 test images (both RGB images of different sizes), that are labeled by 1000 different categories. Classifying images from ImageNet is a nontrivial task even for a human, since this data set contains for example 59 categories for birds of different species and gender [Van Horn et al., 2015]. This may explain why a relaxed performance measurement, where one records whether the target class is among the top 5 classifications that are proposed by the neural network ("Top5"), is typically much higher.

1.2.1 Using FS-coding to emulate EfficientNet with spiking neurons

The recently proposed EfficientNet [Tan and Le, 2019] promises to become a new standard CNN architecture due to its very high accuracy while utilizing a smaller number of parameters than other CNN architectures. EfficientNet uses as activation function $f(x)$ besides the Swish function (Fig. 2) also the familiar sigmoid function, shown as the red curve in Figure 3. Note that 99.97% of its activation functions are Swish functions, making the appearance of the sigmoid function comparatively rare. The Swish function emerged from preceding work on optimizing activation functions in ANNs [Zoph and Le, 2018]. Another characteristic of the EfficientNet architecture is the extensive usage of depth-wise separated convolution layers. In between them, linear activation functions are used. Although it would certainly be possible to approximate linear functions using FS-coding, we simply collapsed linear layers into the generation of the weighted sums that form the inputs to the next layers.

Since the Swish function assumes also negative values, it appears to be difficult to convert an ANN with this activation function via rate-coding to a spiking neuron. But it is fairly easy to convert it to an FS-neuron. The values of the parameters $T(t), h(t)$ and $d(t)$ for $t = 1, ..., K$ of the FS-neuron can be obtained by training the FS-neuron model to fit the Swish function. We used for that backpropagation through time, with a triangle-shaped pseudo derivative for the non-existing derivative of the Heaviside step function.

In most cases, the possible inputs to an activation function are not uniformly distributed, but there exists a certain region in which most inputs lie with a high probability. For example, most of the inputs to the Swish functions in the EfficientNet are in the interval from -2 to 2 and therefore, achieving a high approximation accuracy in this region is especially desirable. It is possible to encourage the FS-neuron to put more emphasis on a certain region, by assigning a high weight in the loss function to this region.

In our experiments, the FS-neurons have been trained to approximate the interval from $[-8, 12]$ for the Swish function and $[-10, 10]$ for the sigmoid function. The resulting FS-neuron approximates the Swish function with a mean squared error of 0.0023 inside the main region $[-2, 2]$ and 0.0064 in the region outside, which can be written as $[-8, -2] \cup [2, 12]$. As a result of our fine-tuning the values for $T(t)$, $d(t)$ and $h(t)$ stay for most time steps t within the main region $[-2, 2]$ as can be seen in Fig 2 B. This supports an iterative processing strategy of the FS-neuron in order to achieve a very good approximation in the main region.

The effective activation function of the resulting FS-neuron is shown in Fig. 2A . Fig. 3

shows the corresponding result for the FS-conversion of an ANN neuron with the sigmoid activation function.

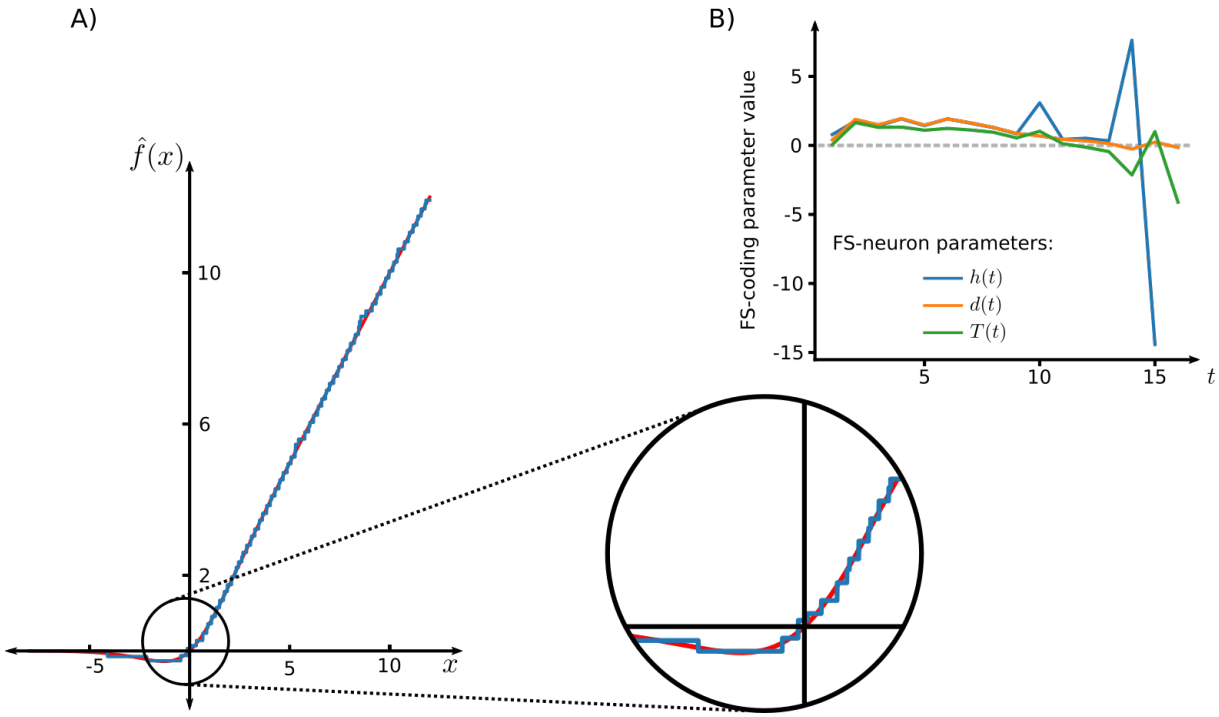

Figure 2: Approximation of the Swish function by an FS-neuron
A) *Approximation quality of the FS-neuron.*
(red: Swish function, blue: FS-approximation with $K = 16$*)*
B) *Optimized internal parameters of the FS-neuron.*

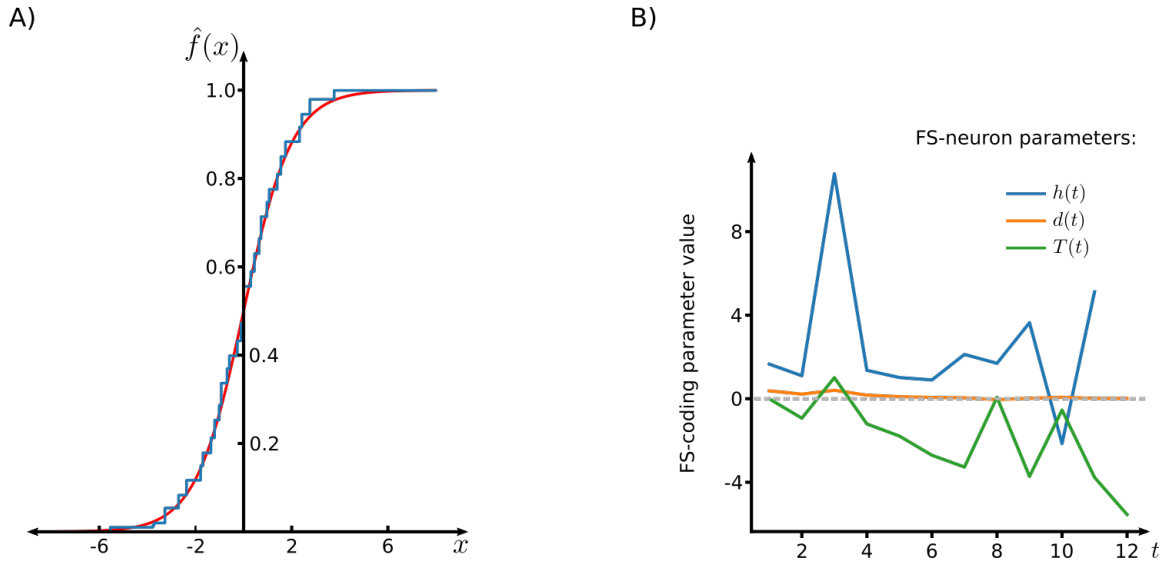

Figure 3: Approximation of the sigmoid function by an FS-neuron
(red: sigmoid function, blue: FS-approximation for $K = 12$).
A) *Approximation quality of the FS-neuron*
B) *Optimized internal parameters of the FS-neuron for $t = 1, ..., 12$.*

Model	# params	ANN	SNN	# layers	# neurons	# spikes
EfficientNet-B7	66M	85% (97.2 %)	83.57% (96.7%)	218	259M	554.9M
ResNet50	26M	75.22% (92.4%)	75.10% (92.36%)	50	9.6M	14.045M

Table 1: Accuracy and spike numbers for classifying images from ImageNet with FS-conversions of two state-of-the-art CNNs. *Top5 accuracy is reported in parentheses. The number of spikes needed for inference was obtained by averaging over the 1000 test images.*

Using these FS-neurons it is possible to emulate the EfficientNet-B7 model with spiking neurons. The accuracy of the resulting spiking CNN, using the publicly available weights w of the trained EfficientNet, can be found in Table 1, together with the total number of spikes that it uses for sample inferences. The FS-conversion of EfficientNet B-7 achieved an accuracy of 83.57%. The best accuracy for ImageNet that had previously been reported for SNNs was 74.6% [Rueckauer et al., 2017]. It was achieved by a rate-based conversion, which required a substantial number of spikes per neuron and about 550 time steps for each image classification. The SNN resulting from FS-conversion of EfficientNet B-7 used about 2 spikes per neuron for classifying an image. The FS-neurons approximating the Swish function used $K = 16$ and the FS-neurons approximating the sigmoid function used $K = 12$. The layers of the CNN that use the Swish function as activation function can

be simulated in a pipelined manner by the SNN, processing a new image every $2K$ time steps: Its first K time steps are spent collecting the outputs from the preceding layer of FS-neurons during their K time steps of activity. It then processes these collected inputs x during the subsequent K time steps. Hence the SNN that results from FS-conversion of EfficientNet can classify a new image every $2K = 32$ time steps.

We decided not to emulate the multiplication operation by FS-neurons, which occurs in the CNN if squeeze and excitation optimization [Hu et al., 2018] is being used. In many neuromorphic chips, such as SpiNNaker and Loihi, the on-chip digital processor could carry out these multiplications. Otherwise one can approximate multiplication in a similar manner as the Swish function with a suitably optimized FS-neuron, see [Stöckl and Maass, 2019]. Alternatively one can compute multiplication with a small circuit of threshold gates, i.e., very simple types of spiking neurons, of depth 2 or 3. A recent summary of such results is provided in section 3 of [Parekh et al., 2018].

1.2.2 Approximating the ReLU activation function

The ReLU activation function, see Fig. 4, is among the most frequently used activation functions, and also quite good accuracies have been achieved with it for ImageNet. It represents a special case for FS-conversion, as it is possible to find the ideal values for $h(t), T(t)$ and $d(t)$ analytically, bases on the idea of computation with binary numbers. By setting the parameters of the FS-neuron to $T(t) = h(t) = d(t) = 2^{K-t}$, the FS-neuron approximates the ReLU activation function $f(x)$ with a coarse-to-fire-processing strategy. Let us assume for simplicity that an FS-neuron receives inputs x from $(-\infty, 0] \cup \{1, 2, ..., 2^K - 1\}$. Then it reproduces with the specified parameters the output $\mathrm{ReLU}(x)$ of the ReLU gate for any x from $(-\infty, 0] \cup \{1, 2, ..., 2^K - 1\}$ without error. In order to be able to transmit also non-integer values x between 0 and some arbitrary positive constant α, one simply multiplies the given values for $T(t), h(t)$ and $d(t)$ with $\alpha 2^{-K}$. Then the FS-neuron reproduces $\mathrm{ReLU}(x)$ for any non-negative x less than α that are multiples of $\alpha 2^{-K}$ without error, and $\mathrm{ReLU}(x)$ is rounded down for values x in between to the next larger multiple of $\alpha 2^{-K}$. Thus the output of the FS-neuron deviates for x in the range from $-\infty$ to α by at most $\alpha 2^{-K}$ from the output of the ReLU gate. The resulting approximation is plotted for $\alpha = 10$ in Fig. 4.

It should be noted that with this choice of parameters of the FS-neuron it is possible to calculate the changes of parameters for $t = 1, ..., K$ by simply using a bit shift operation, possibly providing a very efficient implementation on neuromorphic hardware. The resulting SNN can be used in a pipelined manner, processing a new network input every $2K = 20$ time steps, analogously as for the Swish function. Hence its throughput is substantially better than that of SNNs which result from rate-based ANN-to-SNN conversions of ANNs with the ReLU function, as proposed for example in [Rueckauer et al., 2017, Sengupta et al., 2019]. The Inception-v3 model in [Rueckauer et al., 2017] was reported to yield a SNN that needed 550 time steps to classify an image. Under the assumption that rate-based models profit only very little from pipelining, it is reasonable to estimate that the throughput of an SNN that results from FS-conversion of ReLU gates with $K = 10$

is roughly 25 times higher. The SNN resulting from the rate-based conversion of the ResNet34 model discussed in [Sengupta et al., 2019] has been reported to use 2500 time steps for a classification. Therefore we estimate that the throughput is increased here by a factor around 125 through FS-conversion.

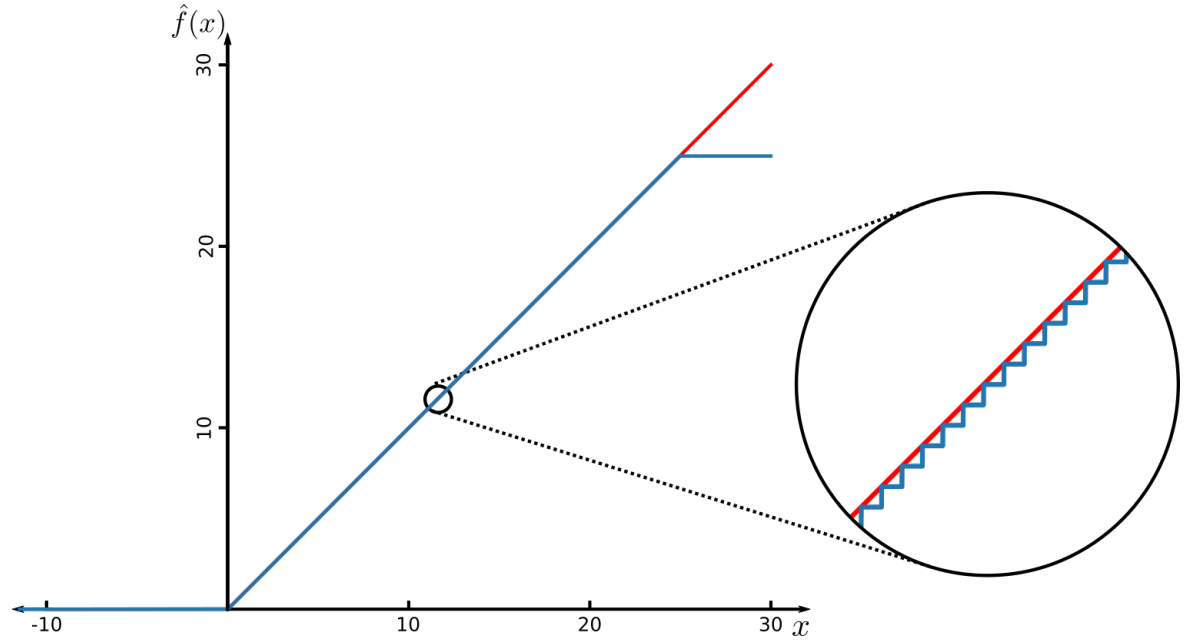

Figure 4: FS approximation of the ReLU function
(red: ReLU function, blue: FS approximation with $K = 10$ and $\alpha = 25$)

The accuracy of 75.22% for the ANN version of ResNet50 in Table 1 resulted from training a variant of ResNet50 where max-pooling was replaced by average pooling, using the hyperparameters given in the TensorFlow repository[†]. The resulting accuracy in ImageNet is close to the best published performance of 76% for ResNet50 ANNs [Tan and Le, 2019, Table 2]. The application of the FS-conversion to this variant of ResNet50 (with $K = 10$ and $\alpha = 25$) yields an SNN whose Top1 and Top5 performance is almost indistinguishable from that of the ANN version.

1.3 Results for the classification of images from the CIFAR10 data set

CIFAR10 [Krizhevsky et al., 2009] is a smaller and more frequently used dataset for image classification. It consists of 60.000 colored images, each having a resolution of just 32 by 32 pixels, and just 10 image classes. The results for ANN versions of ResNet that are given in Table 2 for CIFAR10 arise from training them with the hyperparameters given in the TensorFlow models repository. They use the ReLU function as the only nonlinearity, since

[†]https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

Model	ANN	SNN	# neurons	# spikes
ResNet50	92.99%	92.42%	475.136	647.245
ResNet20	91.58%	91.45%	188.416	261.779
ResNet14	90.49%	90.39%	131.072	190.107
ResNet8	87.22%	87.05%	73.728	103.753

Table 2: Accuracies and spike numbers for classifying images from CIFAR10 with FS-conversions of ResNet models of different depths (using $K = 10$ and $\alpha = 25$). *The number of spikes refers to the average amount of spikes needed for inference over the whole test set.*

we have replaced there max-pooling by average pooling. Nevertheless, they achieve an accuracy for CIFAR10 which is very close to the best results reported for CIFAR10 in the literature. The best performing reported ResNet on CIFAR10 is ResNet110, where a test accuracy of 93.57% had been achieved [He et al., 2016]. Our ResNet50 achieves 92.99%, which is similar to their accuracy of 93.03% forResNet56.

Spiking versions of ResNet20 have already been previously explored [Sengupta et al., 2019]. Using a rate-based conversion scheme an accuracy of 87.46% was reported. FS-conversion of ResNet20 yields a substantially higher accuracy of 91,45%, using just 80 to 500 time steps for each image -depending on the model depth- instead of 2000, thereby significantly reducing latency. In addition, the throughput is drastically improved.

Also the number of spikes that the SNN uses for classifying an image from CIFAR10 is significantly reduced when one moves from a rate-based conversion to an FS conversion. A converted ResNet11 has been reported to use more than 8 million spikes to classify a single test example [Lee et al., 2019]. Comparing this to an FS-converted ResNet14 we find that the latter uses 40 times fewer spikes despite being a slightly larger model. Using direct training of SNNs instead of a conversion scheme has been reported to result in a lower amount of spikes needed to perform a single classification. However, even a directly trained SNN version of ResNet11 uses 7 times more spikes than an FS-conversion of ResNet14 [Lee et al., 2019, Table 8].

1.4 Analysis of the number of spikes needed

On neuromorphic hardware the energy consumption is proportional to the number of spikes which are needed for a computation. The number of spikes needed for an FS-neuron to perform the approximation of the target function is depicted in Figure 5 as function of the gate input x. If one compares these numbers with the distribution of input values x (red curves) that typically occur during image classification, one sees why on average less than 2 spikes are used by FS-neurons for these applications.

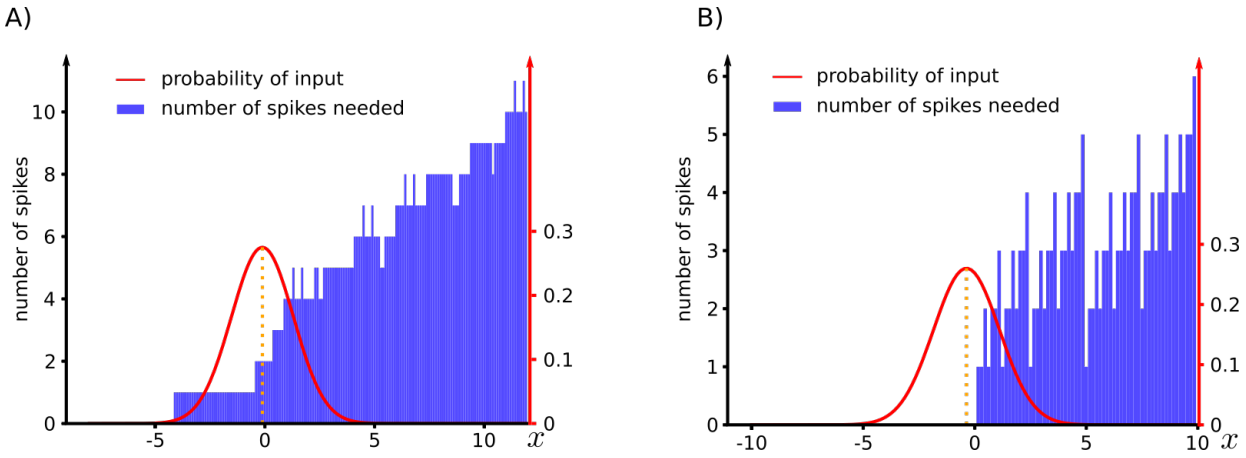

Figure 5: Number of spikes needed by FS-neurons for image classification
A) The number of spikes used by an a FS-neuron with $K = 16$ to approximate the Swish function, as function of its input value x. The red Gaussian models the probability that the FS-neuron will receive this input value in the EfficientNet-B7 model in an application to images from ImageNet (mean $= -0.112$, variance $= 1.99$).
B) The number of spikes used by an FS-neuron to approximate the ReLU function with $K = 6$ and $\alpha = 10$. The red Gaussian models the probability that the FS-neuron will receive this input value in the ResNet50 model in an application to images from ImageNet (mean -0.36970, variance $= 2.19$).

1.5 Impact of the complexity of FS-neurons on the approximation quality

The most important specification of an FS-neuron is the number K of time steps that it uses. Figure 6A, B provide insight into the nature of the trade-off between the size of K and the approximation quality of the FS-neuron.

Furthermore, it is of interest to consider scenarios where only a certain number of bits are available for the FS-neuron parameters. To analyze the impact of that we consider a setting where the parameters of the FS-neurons can only take on discrete values in the range from $[-8, 8]$. The possible values are equally spaced and the number of values can be written as 2^Q, where Q refers to the number of bits which are available for each parameter $T(t)$, $h(t)$, $d(t)$ of the FS-neuron. Figure 6C, D depict the impact of such quantization on the mean squared error of the approximation of the activation function.

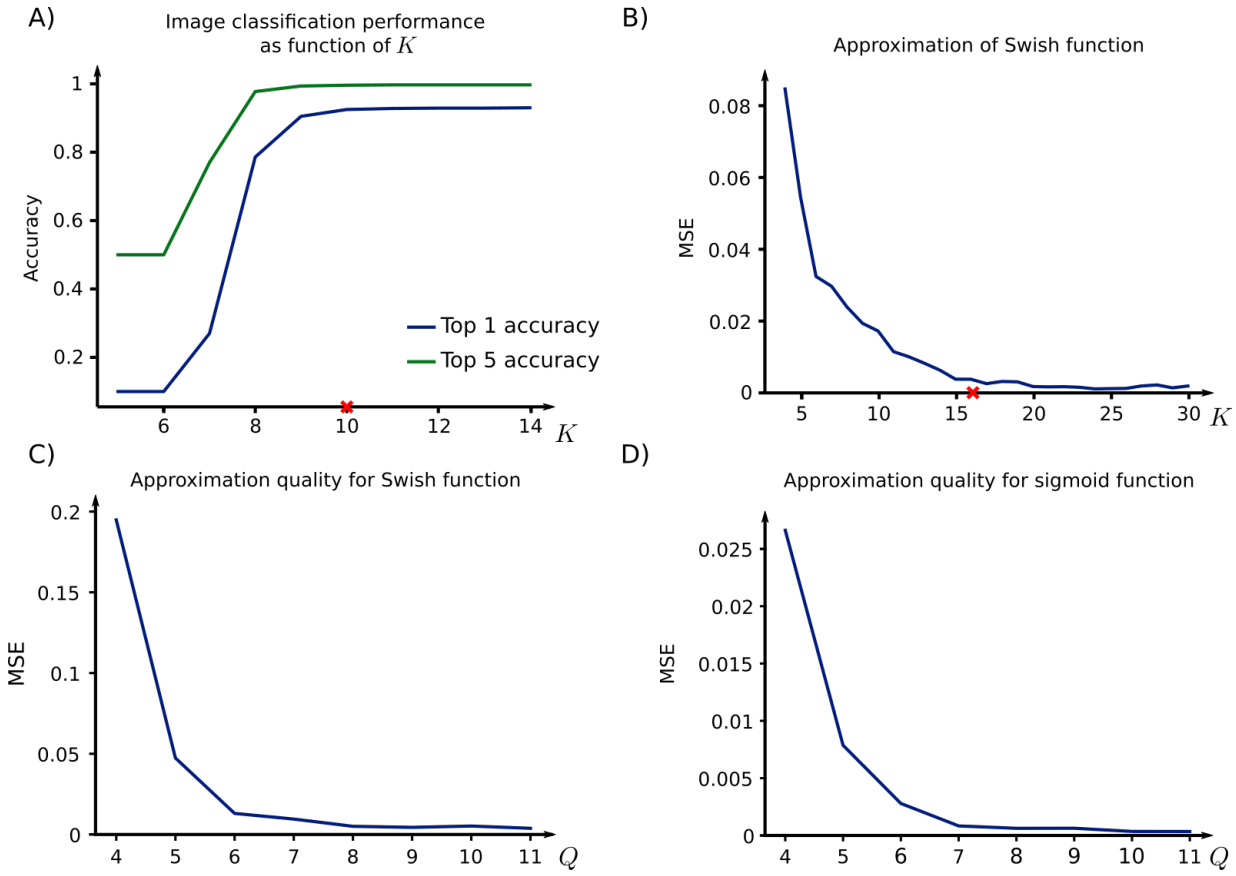

Figure 6: Influence of K and bit precision on performance
A) Test Accuracy of the ResNet50 model on CIFAR10 with FS-neurons, in dependence on K. The red cross indicates the chosen value of K for our results.
B) Mean squared error of a Swish approximation by FS-neurons with different values of K.
C) Mean squared error of a Swish approximation by FS-neurons with $K = 16$ as function of the bit precision of its parameters.
D) Mean squared error of a sigmoid approximation by FS-neurons with $K = 12$ as function of the bit precision of its parameters.

Discussion

We have presented a new approach for generating SNNs that are very close to ANNs in terms of classification accuracy for images, while working in the energetically most attractive regime with very sparse firing activity. Besides substantially improved classification accuracy, they exhibit drastically improved latency and throughput compared with rate-based ANN-to-SNN conversions. Our approach is based on the idea to optimize the spiking neuron model for this purpose, rather than taking a standard model off the shelf. One can

argue that this is exactly the way which evolution has chosen for the design of neurons in living organism. Not only neurons with particular information processing tasks in the smaller nervous systems of insects, but also neurons in the neocortex of mammals exhibit an astounding diversity of genetically encoded response properties [Sterling and Laughlin, 2015, Gouwens et al., 2019, Bakken et al., 2020]. In particular, the probability of producing a spike depends in diverse ways on the recent stimulation history of the neuron, see [Gerstner et al., 2014] for some standard models. In other words, the excitability of different types of biological neurons increases or decreases in complex ways in response to their previous firing. As a result, the temporal structure of a train of spikes that is produced by a biological neuron contains additional information about the neuron input that can not be captured by its firing rate. Similary, FS-neurons that are optimized for high accuracy image classification with few spikes exhibit history-dependent changes -encoded through their functions $T(t)$ and $h(t)$ according to equ. (2)- in their propensity to fire, see Fig. 2B and 3B. Furthermore the function $d(t)$ enables subsequent neurons to decode their spikes in a timing-sensitive manner. In these regards an FS-conversion from ANNs to SNNs captures more of the functional capabilities of spiking neurons than previously considered rate based conversions to an off-the-shelf spiking neuron model.

In particular, the FS-conversion is applicable to that activation function for ANN neurons that currently provides the highest accuracy on ImageNet, the Swish function. Rate-based conversion can not be readily applied to the Swish function because it assumes both positive and negative output values. When approximating the more commonly used ReLU function, FS-neurons approach the information theoretic minimum of spikes for spike-based communication. In fact, FS-neurons that emulate ANN gates with the ReLU activation function produce 1.5 spikes on average for classifying on image, while those for the Switch activation function produce 2 spikes on average. As the number of spikes required for inference by an SNN is directly related to its energy consumption in spike-based neuromorphic hardware, the energy consumption of FS-converted SNNs appears to be close to the theoretical optimum for SNNs. Since FS-conversion provides a tight bound on the number K of time steps during which a spiking neuron is occupied, it can also be used for converting recurrently connected ANNs to SNNs.

The proposed method for generating highly performant SNNs for image classification through FS-conversion of trained CNNs offers an opportunity to combine the computationally more efficient and functionally more powerful training of ANNs with the superior energy-efficiency of SNNs for inference. Note that one can also use the resulting SNN as initialization for further training of the SNN, e.g., for a more specific task.

Altogether our results suggest that spike-based hardware may gain an edge in the competition for the development of drastically more energy-efficient hardware for AI if one does not forgot to optimize the spiking neuron model in the hardware for its intended range of applications. But note that -in contrast to dedicated hardware for specific ANN-algorithms and architectures, resulting SNNs tend to work efficiently for all algorithms and ANN-architectures that use the same types of ANN-neurons. Hence their application range is much broader.

Acknowledgements

We would like to thank Franz Scherr for helpful discussions. This research was partially supported by the Human Brain Project of the European Union (Grant agreement number 785907).

Competing Interests

We are not aware of competing interests.

References

T. E. Bakken, N. L. Jorstad, Q. Hu, B. B. Lake, W. Tian, B. E. Kalmbach, M. Crow, R. D. Hodge, F. M. Krienen, S. A. Sorensen, et al. Evolution of cellular diversity in primary motor cortex of human, marmoset monkey, and mouse. *bioRxiv*, 2020.

G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *bioRxiv 738385*, 2019. doi: 10.1101/738385. URL `http://dx.doi.org/10.1101/738385`.

E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019. ISSN 07437315. doi: 10.1016/j.jpdc.2019.07.007. URL `https://doi.org/10.1016/j.jpdc.2019.07.007`.

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. Neuronal dynamics: From single neurons to networks and models of cognition. *Cambridge University Press*, 2014.

N. W. Gouwens, S. A. Sorensen, J. Berg, C. Lee, T. Jarsky, J. Ting, S. M. Sunkin, D. Feng, C. A. Anastassiou, E. Barkan, et al. Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature neuroscience*, 22(7):1182–1195, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90.

J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00745.

S. R. Kheradpisheh and T. Masquelier. S4NN: temporal backpropagation for spiking neural networks with one spike per neuron. *arXiv preprint arXiv:1910.09495*, 2019. URL `http://arxiv.org/abs/1910.09495`.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

C. Lee, S. S. Sarwar, and K. Roy. Enabling Spike-based Backpropagation in State-of-the-art Deep Neural Network Architectures. *arXive preprint arXiv:1903.06379*, 2019.

J. Ling. `https://hypertextbook.com/facts/2001/JacquelineLing.shtml`, 2001.

W. Maass and T. Natschläger. Emulation of Hopfield networks with spiking neurons in temporal coding. In *Computational Neuroscience*, pages 221–226. Springer, 1998.

O. Parekh, C. A. Phillips, C. D. James, and J. B. Aimone. Constant-depth and subcubic-size threshold circuits for matrix multiplication. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 67–76, 2018.

B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11(DEC):1–12, 2017. ISSN 1662453X. doi: 10.3389/fnins.2017.00682.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. ISSN 15731405. doi: 10.1007/s11263-015-0816-y. URL `http://dx.doi.org/10.1007/s11263-015-0816-y`.

A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in Neuroscience*, 13(1998):1–16, 2019. ISSN 1662-453X. doi: 10.3389/fnins.2019.00095.

P. Sterling and S. Laughlin. *Principles of neural design*. MIT Press, 2015.

C. Stöckl and W. Maass. Recognizing images with at most one spike per neuron. *arXiv preprint arXiv:2001.01682*, 2019.

M. Tan and Q. V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946*, 2019.

S. Thorpe, A. Delorme, and R. Rullen. Spike-based strategies for rapid processing. *Neural networks : the official journal of the International Neural Network Society*, 14:715–25, 07 2001. doi: 10.1016/S0893-6080(01)00083-1.

G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 595–604, 2015.

B. Zoph and Q. V. Le. Searching for activation functions. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, pages 1–13, 2018.