EFFICIENCY OF SPIKING NEURAL NETWORKS

DRAFT

Daniel Niederlechner
Project in Computer Science
Logic and Computation
TU Vienna

July 17, 2019

ABSTRACT

Artificial Neural Networks became quite popular during the past two decades, due to better training algorithms and the tremendous increase in computational power. These networks, however, represent only a very simplified version of their biological model. More recently, architectures, which try to more precisely mimic biological neurons by also modeling temporal encoding and sequences of spikes, are attracting more and more attention. The resulting networks are called Spiking Neural Network, and have a great potential for solving complex problems like object recognition with low power consumption on specialized neuromorphic hardware. This work is focused on the differences in efficiency of spiking neural networks compared to their analog counterparts.

***K*eywords** Convolution · Spiking Neurons · Deep Learning

1 Introduction

A spiking neural network (SNN) is fundamentally different from classical artificial neural networks. SNNs model temporal events called spikes, rather than just computing continuous values, to propagate through the network. The occurrence of a spike is determined by a membrane function, simulating the threshold and potential of the neuron. Once a neuron reaches a certain threshold, it releases spikes and resets its potential. There already exists a variant of different tools for building and efficiently simulating these networks. In this work, different convolutional neural networks (CNN) are build and trained to solve basic image classification tasks. These models are then converted into spiking neural networks, by relying on the results and tools of some of the recently published work on this topic. Metrics for effectiveness and efficiency are then calculated and used to compare the results of ANNs and SNNs, followed by thoughts and ideas for improvements and topics for further investigations.

2 Experiment Setup

A total of 8 different CNN Models, inspired by state-of-the-art architectures (e.g. LeNet), on 2 different Datasets, were built, trained and tested with the help of the keras API (using TensorFlow as back-end). One goal was to achieve decent benchmark scores, with as little variation as possible among models concerning the same data set. These models were then converted into spiking neural networks using the ***snntoolbox*** (`https://snntoolbox.readthedocs.io/en/latest/guide/intro.html`). Throughout the training-, convertion- and simulation process, several graphical and numerical statistics based on model- effectiveness and efficiency were collected and analyzed afterwards. A large proportion of subsequent calculations are based on suggestions and results in [1].

Hardware Specifications: Processing was done on a Intel i5-8300H CPU (2.30GHz x 8) accompanied with GPU processing on a GeForce GTX 1050 Ti whenever possible. Up to 16 GB RAM were used for the simulations of the spiking neural networks.

2.1 Data Sets and Preprocessing

The first half of the models were trained and evaluated on the MNIST database of handwritten digits, including a training set of 60,000 examples, and a validation set of 10,000 examples of 28 x 28 black and white images in 10 classes. The pixel values of the images range between 0 and 255 by default, hence they were normalized.

The rest of the models were trained and evaluated on the the CIFAR-10 dataset, containing 60,000 32x32 color images in 10 different classes. The images were split into a 50,000 example training set and a 10,000 example test set. The pixel values were also normalized. In contrast to the MNIST data set, here all images come with 3 color channels instead of 1.

2.2 Convolutional Neural Network Architecture

The basic architectures, in addition to layer size, are only varying in the number of convolution layer bundles and dense layers. There are two different types of convolutional layer bundles:

- Bundles of Type I are composed of a Conv2D-, a MaxPooling- and a Dropout(CNN)-layer
- Bundles of Type II are composed of two consecutive Conv2D-layers, followed by a MaxPooling- and a Dropout(CNN)-layer

Kernels and Strides are the same throughout all models. The convolution layer bundles are followed by a Flatten-layer and some variation of Dense- and Dropout-layers. (for more information on the CNN-architectures, see `https://github.com/da-ni/efficiency-of-snn/blob/master/models/ANN/models_summary.txt`)

Table 1: Differences in Network Architectures

	Trainable Layers			
Name	Total	Conv2D	Dense	Trainable Parameter
SmallCNN1	2	1	1	347,146
MediumCNN1	3	2	1	356,394
LargeCNN1	5	4	1	136,238
DeepCNN1	7	4	3	70,410
SmallCNN2	2	1	1	3,691,018
MediumCNN2	3	2	1	3,727,946
LargeCNN2	5	4	1	1,250,858
DeepCNN2	7	4	3	1,411,242

2.3 Conversion to Spiking Neural Networks

Training spiking neural networks is difficult, due to the non-differentiable nature of spike signals. The lack of differentiable activation functions prevents the use of backpropagation, which is the most common algorithm for training deep neural networks (although there are attempts to solve this issue, e.g. [3]). An alternative approach is to train an analog (i.e. real-valued activations) neural network (ANN), use its weights and convert the analog neurons by simple integrate-and-fire spiking neurons. The obtained network can then be run on a simulator.

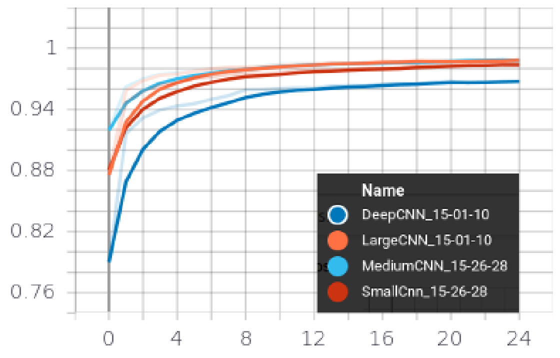
For this experiment, the keras-based INIsim simulator (built-in ***snntoolbox***) was used, since it is the only simulator which currently supports convertion of Max-Pooling and softmax activation layers. For every network, the parameters of each layer were normalized by the 99.9-percentile, since this significantly reduces the time-steps needed for the network to converge to its error threshold [1]. Since the general approach was to always exhaust the 16GB RAM limit, other configuration parameter of the simulation, such as number of time-steps, input batch-size and the total number of input images to test on, vary among different network simulations.

3 Results

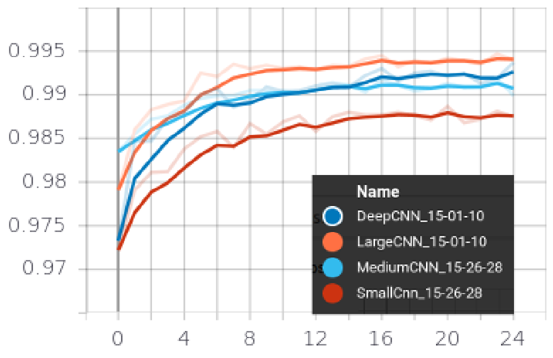
This section presents the most relevant findings obtained by the training of ANNs, simulation of the converted SNNs and the comparison of efficiency and effectiveness measures. A collection of all recorded results is available at `https://github.com/da-ni/efficiency-of-snn/tree/master/results`.

3.1 Training of Convolutional Neural Networks

For the MNIST data set, different models were trained and validated using a 85/15 train-validation split. All models were trained for 25 epochs. Without further techniques or data augmentation, the achieved mean accuracy score of 99.3% represents a decent result compared to common benchmarks. (for comparison, see `https://benchmarks.ai/mnist`)



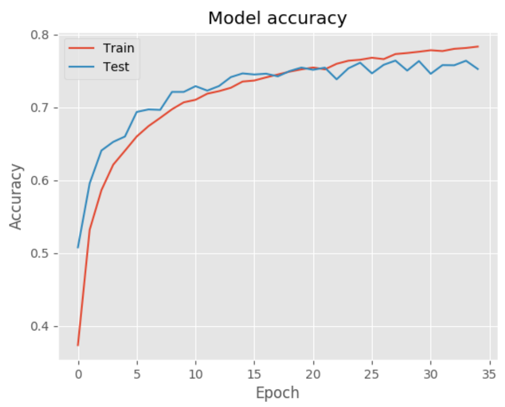
(a) Training Accuracy



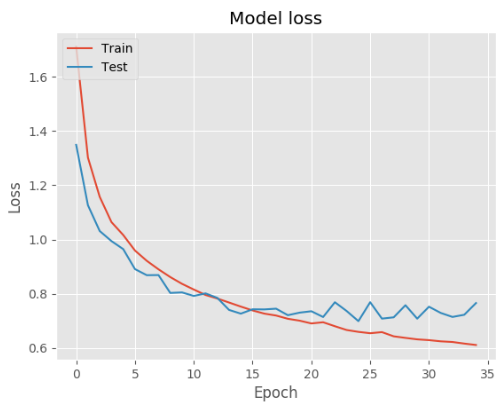
(b) Validation Accuracy

Figure 1: Tensorboard session of the 4 best models on MNIST

For the CIFAR10 data set, different models were trained and validated using a 83/17 train-validation split. At first, all models were trained for 35 epochs. However, there were some problems with respect to over-fitting. (see figure 2). To compensate for this, the number epochs during training was adjusted to each model accordingly. The achieved mean accuracy score of 72% represents an acceptable results, considering the small amount of training epochs.



(a) Accuracy of Model of type LargeCNN

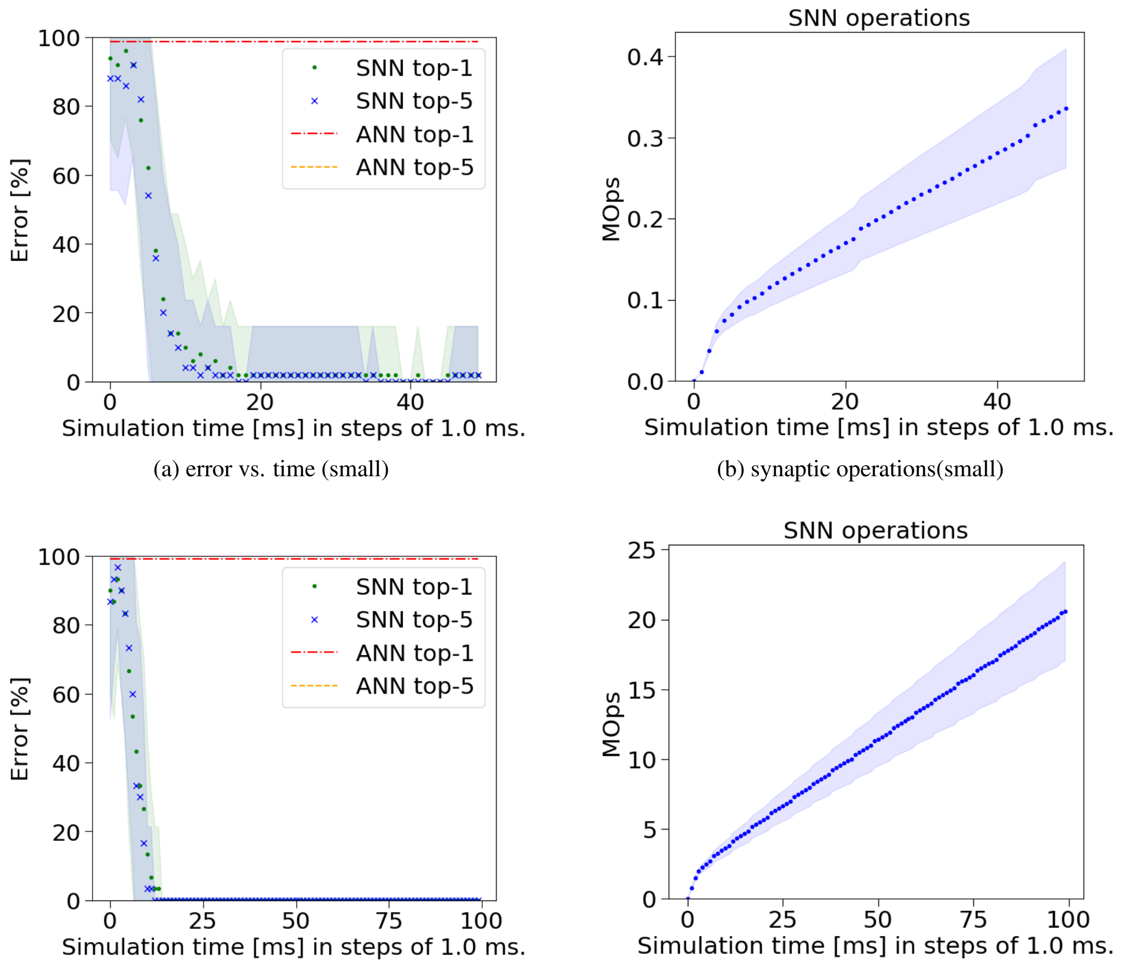


(b) Loss of Model of type LargeCNN

Figure 2: Example of model over-fitting after 35 epochs

3.2 Simulation of Spiking Networks

It was possible to convert all models on the MNIST data set without a loss in effectiveness (see table 2). In total, the models were tested on 300-500 images in badges of size 30-50. The duration of the simulations was set to 100ms (1ms $\simeq$ 1 step), although the error already converged after 50 ms or even faster for smaller models (figure 3 (a), (c)). It is also observable, that the number of synaptic operations grows very fast initially, and continues to grow linearly after a few time-steps. This seems to represents the time needed for input spikes to propagate through network, i.e. to "fully activate" all the layers (figure 3 (b), (d))



(a) error vs. time (small)

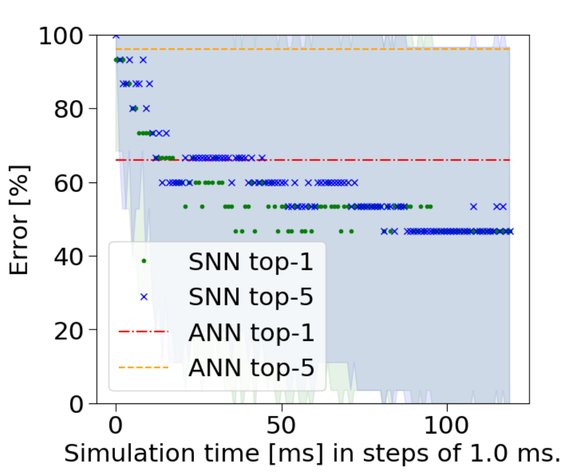
(b) synaptic operations(small)

(c) error vs. time (medium)

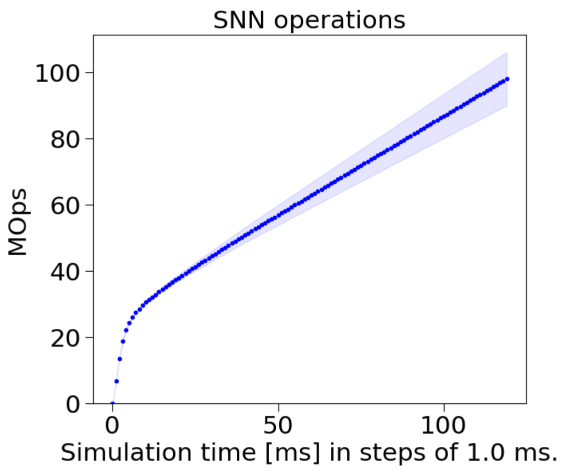
(d) synaptic operations (medium)

Figure 3: Simulation results of models SmallCNN1(top) and MediumCNN2(bottom) on MNIST data set

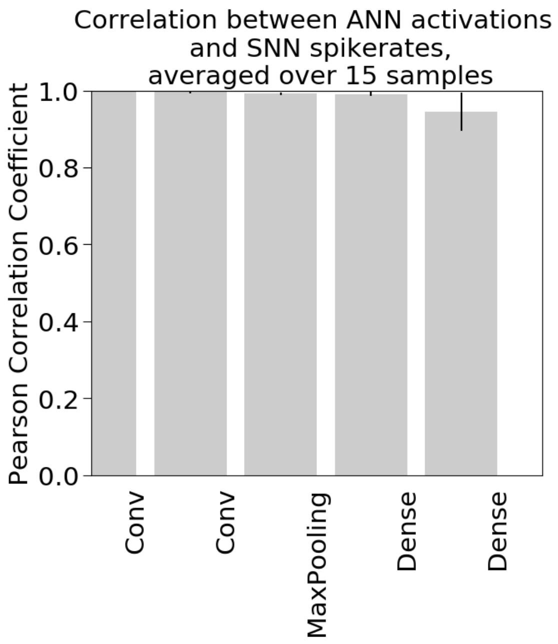
The simulation of the models on the CIFAR-10 data set was done on 150-200 test images in badges of 15-20. The duration of the simulations was set to 100-150ms (1ms $\simeq$ 1 step). The error rate of the simulated converted models was on average 10% higher compared to their ANN counterparts. The convergence of the error rate is also much slower compared to the previous simulations on the MNIST data set, despite the high correlation of activations and spike-rates (figure 4). One possible explanation might be the dynamic event-driven computations of spiking networks are not suitable for classical convolutional- and dense-layer architectures (compare section 3.4). The authors of the ***snntoolbox*** also mention that the spiking softmax activation tends to reduce accuracy due to the nature of its implementation.



(a) error vs. time (medium)



(b) synaptic operations (medium)



(c) correlation of ANN and SNN (medium)

Figure 4: Simulation results of model MediumCNN2 on CIFAR-10 data set

Table 2 compares all results so far. The number of flops (here: floating point operations), the number of multiply-accumulate operations and the total costs were calculated post-hoc as suggested in [1]. The number of synaptic operations were measured during the simulation and accumulated up to the first time step reaching the error threshold. The Factor column represents the ratio of Total-Costs to SO-Costs.

Table 2: Comparing results of ANN and SNN

		CNN			SNN		
Name	Dataset	Acc.[%]	MFlops*	Total-Costs**	Acc.[%]	SO-Costs***	Factor
SmallCNN1	MNIST	98.75	0.80	6.030	98.80	0.1-0.15	46.38
MediumCNN1	MNIST	98.99	1.10	8.239	99.00	2.5-3	3.05
LargeCNN1	MNIST	99.37	0.43	3.245	98.67	7.5-8	0.42
DeepCNN1	MNIST	99.40	0.49	3.642	98.33	15-18	0.22
SmallCNN2	CIFAR10	69.75	7.43	55.708	59.50	3-8	11.14
MediumCNN2	CIFAR10	74.63	8.47	63.542	66.00	100-120	0.58
LargeCNN2	CIFAR10	75.27	3.27	24.528	49.33	60-70	0.38
DeepCNN2	CIFAR10	65.77	3.49	26.201	51.50	120-150	0.19

* Million Floating Point Operations, calculated as suggested in [1]. This number is constant in classic ANN

** Multiply–Accumulate operation costs are estimated 14X higher than (artificial) synaptic operation costs. (for further details also see [1]). Here, half of all Flops are MAC.

*** In SNN, the number of synaptic operations is dynamic and increases with every time-step. Usually, the longer a continuous input is present, the higher the accuracy (latency-accuracy trade-off)

3.3 Interpretation of the Results

The shallow spiking neural networks (up to 3 hidden layers) were especially efficient . For example, the converted SmallCNN1 model, with its 347,146 trainable parameter, was 46X more efficient than its ANN counterpart. Also the larger model SmallCNN2, including 10 times as much parameters, was still 11X more efficient than its analog variant. However, by adding more convolutional and dense layers, this factor decreases rapidly.
When comparing the factors of the LargeCNN and DeepCNN architectures, it is noticeable that there is no significant difference between the MNIST and CIFAR-10 results. Their factor seems to be stable, even when number of trainable parameter in the network increases one order of magnitude. This also strengthens the assumption that, instead of the number of neurons, it is the depth of the network, that has the main impact on the efficiency of SNN.

3.4 Topics for Further Investigation

Some of the recent work is focused on sparse computation in neural networks. The efficiency of spiking neural networks in the brain, for example, largely derives from its sparse activity. In contrast, current state-of-the-art SNNs use stochastic Poisson neurons with high firing rates to simulate the corresponding analog neurons. One possible way to reduce this firing rates is presented in [4]. Network architectures which make use of sparse representations are definitely worth considering regarding network robustness and efficiency.

In [5], the authors present the *Thousand Brains Theory of Intelligence*, an attempt to describe how the neocortex works. Rather than learning one model of an object (or concept), they propose that the brain builds many models, each built using different inputs. The models then vote together to reach a consensus on what to predict. Following their idea, it might be interesting to implement a homogeneous ensemble of shallow neural networks. According to the results presented in table 2, a SNN implementation of this small and shallow networks could turn out to be a very efficient model for classification tasks.

4 Conclusion

To investigate the efficiency of spiking neural networks, different convolutional analog models, varying in depth, layer size and basic architecture, were trained on the MNIST and the CIFAR-10 data set and afterwards converted with the help of ***snntoolbox***. The properties of the converted SNNs were then studied using the INIsim simulator. By comparing floating-point operation of the classic ANNs and synaptic operations of the SNNs, a 10-45 fold increase in efficiency was observed for the smallest tested network architecture. With every additional convolution- and dense layer, this factor seemed to drop rapidly. In general, additional convolutional layers had more impact on the efficiency than dense layers, and the depth of the network had more impact on the efficiency than the total number of neurons. Taking this results into account, two suggestions were stated for possible improvements of spiking neural network architectures : layers with sparse computations/representations and a homogeneous ensemble of small, shallow neural networks.

References

[1] Rueckauer, B. and Hu, Y. and Lungu, I.A. and Pfeiffer, M. and Liu, S.-C.
Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,
Front. Neurosci., 2017, doi: 10.3389/fnins.2017.00682

[2] Zambrano Davide, Nusselder Roeland, Scholte H. Steven and Bohté Sander M.
Sparse Computation in Adaptive Spiking Neural Networks
Frontiers in Neuroscience, Vol.12, 2019, pg. 987,
https://www.frontiersin.org/article/10.3389/fnins.2018.00987, doi:10.3389/fnins.2018.00987, issn:1662-453X

[3] Lee Jun Haeng, Delbruck Tobi, Pfeiffer Michael
Training Deep Spiking Neural Networks Using Backpropagation
Frontiers in Neuroscience, Vol.10, 2016, pg.508 ,doi:10.3389/fnins.2016.00508, issn:1662-453X

[4] Zambrano Davide, Nusselder Roeland, Scholte H. Steven, Bohté Sander M.
Computation in Adaptive Spiking Neural Networks
Frontiers in Neuroscience, Vol.12,2019, pg.987, doi:10.3389/fnins.2018.00987, issn:1662-453X

[5] Jeff Hawkins, Marcus Lewis, Scott Purdy, Mirko Klukas, Subutai Ahmad
A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex
Frontiers in Neural Circuits 12, 121. doi:10.3389/FNCIR.2018.00121