Theory and Tools for the Conversion of Analog to Spiking Convolutional Neural Networks

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer
Institute of Neuroinformatics
University of Zurich and ETH Zurich
Winterthurerstrasse 190
8057 Zurich, Switzerland
`{rbodo,iulialexandra,yuhu,pfeiffer}@ini.uzh.ch`

Abstract

Deep convolutional neural networks (CNNs) have shown great potential for numerous real-world machine learning applications, but performing inference in large CNNs in real-time remains a challenge. We have previously demonstrated that traditional CNNs can be converted into deep spiking neural networks (SNNs), which exhibit similar accuracy while reducing both latency and computational load as a consequence of their data-driven, event-based style of computing. Here we provide a novel theory that explains why this conversion is successful, and derive from it several new tools to convert a larger and more powerful class of deep networks into SNNs. We identify the main sources of approximation errors in previous conversion methods, and propose simple mechanisms to fix these issues. Furthermore, we develop spiking implementations of common CNN operations such as max-pooling, softmax, and batch-normalization, which allow almost loss-less conversion of arbitrary CNN architectures into the spiking domain. Empirical evaluation of different network architectures on the MNIST and CIFAR10 benchmarks leads to the best SNN results reported to date.

1 Introduction

Computation in Spiking Neural Networks (SNNs) is event-based and data-driven, and thus neurons only update whenever new relevant information needs to be processed. This contrasts with the frame-based computations performed by traditional Analog Neural Networks (ANNs), which process the entirety of the input, followed by computing all neuronal activations layer-by-layer before producing a final output. Recent results such as (Neil et al., 2016; Farabet et al., 2012; O'Connor et al., 2013; Zambrano and Bohte, 2016) have shown that the event-based mode of operation in SNNs is particularly attractive for reducing latency and computational load in deep neural networks, which represent the state of the art in most machine learning benchmarks (LeCun et al., 2015). Deep SNNs can be queried for results already after the first output spike is produced, unlike ANNs where the result is available only after all layers have been completely processed (Diehl et al., 2015). SNNs are also naturally suited to process input from event-based sensors (Liu et al., 2014; Posch et al., 2014), but even for classical frame-based machine vision applications such as object recognition or detection, SNNs have been shown to be accurate, fast, and efficient, in particular when run on neuromorphic hardware platforms (Esser et al., 2016; Neil and Liu, 2013; Stromatias et al., 2015). SNNs could thus play an important role in supporting, or in some cases replacing deep ANNs in tasks where fast and efficient classification in real-time is crucial, such as detection of objects in larger and moving scenes, tracking tasks, or activity recognition (Hu et al., 2016).

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Training Deep SNNs directly from their spiking activity is notoriously difficult, and only recently methods for backpropagation-like training of SNNs have been developed (Lee et al., 2016). However, a number of studies have shown that SNNs can be successfully constructed by converting conventionally trained ANNs, relating the activations of ANN units to firing rates of spiking neurons. The purpose of this study is to identify a number of challenges in the conversion of networks through a novel theory, and proposing new mechanisms which significantly improve the performance of deep SNNs.

Previous work on ANN-to-SNN conversion starts with (Perez-Carrasco et al., 2013), where CNN units were translated into biologically inspired spiking units with leaks and refractory periods, aiming for processing of inputs from event-based sensors. (Cao et al., 2015) suggested a close link between the transfer function of a spiking neuron, i.e. the relation between input current and output firing frequency to the activation of a rectified linear unit (ReLU), which is nowadays the standard model for units in ANNs. They achieved good performance on conventional computer vision benchmarks, converting a class of CNNs that was restricted to having zero bias and only average-pooling layers. Their method was improved by (Diehl et al., 2015), who achieved nearly loss-less conversion of ANNs for the MNIST task through *weight normalization*. This technique rescales the weights to avoid approximation errors in SNNs due to either excessive or too little firing. (Hunsberger and Eliasmith, 2015) introduced a conversion method where noise injection during training improves the robustness to approximation errors of the SNN with biologically more realistic neuron models. (Esser et al., 2016) demonstrated an approach that optimized CNNs for the neuromorphic TrueNorth platform with low-precision weights and restricted connectivity. Recently, (Zambrano and Bohte, 2016) have developed adapting SNNs, which encode information with a minimum number of spikes, thereby achieving good accuracy with an order of magnitude less spikes than in other SNN approaches. All these approaches showed that SNNs can achieve near state-of-the-art accuracy, while at the same time improving classification speed, and aiming towards constructing deep SNNs that can run on low-power neuromorphic platforms.

In this work we investigate in detail the conversion algorithm proposed in (Diehl et al., 2015), which can convert small ANN models into SNNs with minimal accuracy loss (less than 1% on MNIST). However, we found that running the same algorithms without modifications on larger networks (e.g. for CIFAR10) leads to an unacceptable drop in accuracy on the order of 10% or more. In our tests we identified that the main reason for a drop of performance is a reduction of firing rates in higher layers, which can arise due to overly pessimistic weight normalization. Furthermore, the algorithms proposed by (Diehl et al., 2015) and (Cao et al., 2015) had severe restrictions on what kind of CNNs could be converted. This prevented using features commonly used in the most successful CNNs for visual classification, such as max-pooling, softmax, batch normalization (Ioffe and Szegedy, 2015), or even basic features of neuron models such as biases. In the following we first analytically investigate the problem of ANN-to-SNN conversion in Section 2, before introducing a set of new tools and tricks that expand the class of networks that can be converted and improve accuracy (Section 3). The contribution of the proposed methods is evaluated on the CIFAR10 benchmark in Section 4.

2 Theory of Conversion of ANNs into SNNs

In this section we investigate analytically how firing rates in SNNs approximate ReLU activations in ANNs. This was suggested first by (Cao et al., 2015) as the basis of ANN-to-SNN conversion, but a theoretical basis for this principle so far has been lacking. From the basic approximation equations we derive a simple modification of the reset mechanism after spikes, which turns each SNN neuron into an unbiased approximator of the target function. We assume here a one-to-one correspondence between ANN units and SNN neurons, even though it is also possible to represent each ANN unit by a population of spiking neurons. For a network with L layers let $\mathbf{W}^l, l \in \{1, \dots, L\}$ denote the weight matrix connecting units in layer $l-1$ to layer l, with biases $\mathbf{b}^l$. The number of units in each layer is M^l. In the ANN, ReLU activations of neuron i in layer l are computed according to

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

starting with $\mathbf{a}^0 = \mathbf{x}$ as the input, which is assumed to be normalized so that each $x_i \in [0, 1]$.

Each SNN neuron has a membrane potential $V_i^l(t)$, which is driven by the input current

$$z_i^l(t) := \tau \left(\sum_{j=1}^{M^{l-1}} W_{ij}^l \Theta_{t,j}^{l-1} + b_i^l \right) \quad , \tag{2}$$

where $\Theta_{t,i}^l$ is a step function indicating the occurrence of a spike at time t:

$$\Theta_{t,i}^l := \Theta(V_i^l(t-1) + z_i^l(t) - \tau), \text{ with } \Theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else.} \end{cases} \tag{3}$$

Every input pattern is presented for T time steps, with time step $dt \in \mathbb{R}^+$. The highest firing rate supported by the simulator is given by the inverse time resolution $r_{\max} := 1/dt$, and input rates are proportional to the pixel intensity or RGB value. We can compute the firing rate of each SNN neuron as $r_i^l(T) := N_i^l(T)/T$, where $N_i^l(T) := \sum_{t=1}^T \Theta_{t,i}^l$ is the number of spikes generated. The principle of ANN-to-SNN conversion, as introduced in (Cao et al., 2015; Diehl et al., 2015) is that the firing rates r_i^l should correlate with the original ANN activations a_i^l in (1) such that $r_i^l(T) \to a_i^l r_{\max}$. This relationship is formalized by introducing a membrane equation for each spiking neuron, and estimating the mean firing rates $r_i^l(T)$ in a next step.

Membrane equation The spiking neuron integrates inputs $z_i^l(t)$ until the membrane potential $V_i^l(t-1)$ exceeds a threshold $\tau \in \mathbb{R}^+$ and a spike is generated. At that time, the membrane potential is reset, and here we compare two types of reset: *reset to zero*, which is used in (Diehl et al., 2015) always sets the membrane potential back to a baseline, typically zero, whereas *reset by subtraction* subtracts the threshold τ from the membrane potential at the time where the threshold is exceeded:

$$V_i^l(t) = \begin{cases} \left(V_i^l(t-1) + z_i^l(t)\right)\left(1 - \Theta_{t-1,i}^l\right) & \text{reset to zero} \quad (4a) \\ V_i^l(t-1) + z_i^l(t) \quad - \tau\Theta_{t-1,i}^l & \text{reset by subtraction} \quad (4b) \end{cases}$$

From these membrane equations we can derive slightly different approximation properties for the two reset mechanisms. For simplicity we first assume that the input currents $z_i^1 = \tau a_i^1 > 0$ remain constant over time, that $V_i^l(0) = 0$, and analyze only the first hidden layer. In the *reset to zero* case this implies that there will always be a constant number of time steps n between spikes of the same neuron, and the threshold will always be exceeded by the same constant amount $\epsilon_i^1 = V_i^1(n) - \tau = n \cdot z_i^1 - \tau$. The firing rate of this neuron $r_i^1(t)$ will therefore be $r_{\max}/n$, but one can easily derive that in an exact non-time stepped simulation, the threshold would have been crossed earlier, namely at $t^* = n \cdot \frac{\tau}{\tau + \epsilon_i^1}$. Given that $\tau + \epsilon_i^1 = V_i^1(n) = n \cdot \tau \cdot a_i^1$ we see that $r_{\max}/t^*$ would be the correct estimate of the firing rate, and the reset to zero mechanism inevitably leads to an approximation error, which appears as the second term on the right in

$$r_i^1(t) = a_i^1 r_{\max} - r_{\max} \frac{\epsilon_i^1}{n \cdot \tau}. \tag{5}$$

This error depends mainly on ϵ_i^1 and does not go away simply with longer simulation time. For shallow networks and easy tasks such as MNIST this error seems to be a minor problem, but we have found that an accumulation of approximation errors in deeper layers degrades the accuracy. We also see from (5) that larger τ and smaller inputs improve the approximation, at the expense of longer integration times. This is because ϵ_i^1 is bounded by z_i^1 if $n > 1$, and also because n will increase for smaller inputs. This is further explanation why the weight normalization scheme proposed in (Diehl et al., 2015), according to which network activations never exceed unity improves performance in the reset-to-zero case. Another obvious possibility to improve the approximation is to reduce the simulation time step, but this comes at the cost of increased computational effort.

A simple switch to the *reset by subtraction* mechanism improves the approximation, and makes the conversion scheme suitable also for deeper networks. In this case the time between spikes is not constant, and the membrane potential at any time point is given as $V_i^1(T) = T \cdot z_i^1 - N_i^1(T) \cdot \tau$. From this we get $N_i^1(T) = \left\lfloor \frac{T \cdot z_i^1 - V_i^1(T)}{\tau} \right\rfloor$, and can estimate

$$r_i^1(T) = \frac{N_i^1(T)}{T} = \frac{z_i^1}{\tau} - \frac{V_i^1(T)}{\tau T} = a_i^1 r_{\max} - \frac{1}{\tau T} V_i^1(T). \tag{6}$$

This means that the firing rate estimate converges to its target value $a_i^1 \cdot r_{\max}$, with the only approximation error due to the discrete sampling. This mechanism therefore leads to more accurate approximations of the underlying ANN than in the methods proposed in (Cao et al., 2015; Diehl et al., 2015). We will show later in Section 4 that this results in improved accuracy in larger networks. A potential problem exists in the case where $z_i^1 > \tau$. In this case the best a neuron can do is to fire with its maximal firing rate $r_{\max}$, but can never fully reach its target frequency. This again makes the case for using weight normalization as in (Diehl et al., 2015), in order to prevent saturation of firing.

Firing rates in higher layers The previous results were based on the assumption that the neuron receives a constant input z at each step of the simulation. When using spiking neurons in the hidden neurons, this condition only holds for the first hidden layer and input in the form of analog currents instead of irregular spike trains. For the *reset-by-subtraction* case we can analytically derive how the approximation error propagates through the deeper layers of the network. For this we insert the expression for SNN input z_i^l from (2) into the membrane equation (4b) for $l > 1$, average $V_i^l(t)$ over the simulation time T, and solve for the firing rate $r_i^l(T)$. This yields

$$r_i^l(T) = \sum_{j=1}^{M^{l-1}} W_{ij}^l r_j^{l-1}(T) + r_{\max} b_i^l - \frac{V_i^l(T)}{\tau T}. \tag{7}$$

This result is not surprising, since the firing rate of a neuron in layer l is given by the weighted sum of the firing rates of the previous layer, minus the time-decaying approximation error that was also found in the first layer. This shows that the approximation errors of earlier layers are propagated through the network, and multiplied with the weights of the next higher layer. The recursive expression (7) can be solved iteratively by inserting the rates of the previous layer rates, starting with the known rates of the first layer (6):

$$r_i^l = a_i^l r_{\max} - \Delta V_{i_l}^l - \sum_{i_{l-1}=1}^{M^{l-1}} W_{i_l i_{l-1}}^l \Delta V_{i_{l-1}}^{l-1} - \cdots - \sum_{i_{l-1}=1}^{M^{l-1}} W_{i_l i_{l-1}}^l \cdots \sum_{i_1=1}^{M^1} W_{i_2 i_1}^2 \Delta V_{i_1}^1 \tag{8}$$

with $\Delta V_i^l := V_i^l(T)/(\tau T)$. Thus, a neuron i in layer l receives an input spike train with a slightly lower spike rate, reduced according to the sampling error ΔV of previous layer neurons. These errors accumulate for higher layers, which explains why it takes longer to achieve high correlations of ANN activations, and why SNN firing rates deteriorate in higher layers.

3 New Methods for ANN-to-SNN Conversion

In the following we introduce new methods and heuristics that improve the classification accuracy of deep SNNs, by either allowing the conversion of a wider ranger of ANNs, or by reducing approximation errors in the SNN.

3.1 Converting biases

Biases are standard in ANNs, but were explicitly excluded by previous conversion methods for SNNs. In a spiking network, a bias can simply be implemented with a constant input current (proportional to the ANN bias) to each cell's membrane potential. The theory in Section 2 fully applies to the case of neurons with biases, and the following Section 3.2 shows how parameter normalization can be applied to biases as well.

3.2 Parameter normalization

One source of approximation errors is that in an SNN in time-stepped simulation neurons are restricted to a firing rate range of $[0, r_{\max}]$, whereas ANNs have no such constraints. (Diehl et al., 2015) have introduced weight normalization as a means to avoid approximation errors due to too low or too high firing, thereby significantly improving the performance of converted SNNs. Here we extend the *data-based weight normalization* mechanism introduced in (Diehl et al., 2015) to the case of neurons with biases and suggest a heuristic that makes the normalization process more robust to outliers.

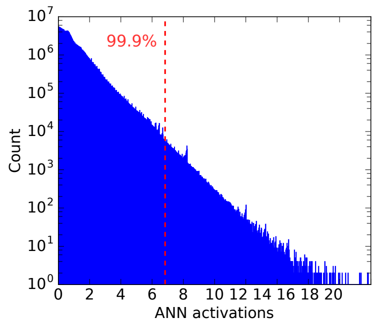
3.2.1 Normalization with biases

The *data-based normalization* scheme from (Diehl et al., 2015) is based on the linearity of the ReLU unit used for ANNs. It can simply be extended to biases by linearly rescaling all weights such that the ANN activation a is smaller than 1 for all training examples. In order to preserve the information encoded within a layer, the parameters of a layer need to be scaled jointly. Denoting the maximum ReLU activation in layer l as $\lambda^l = \max[\mathbf{a}^l]$, then weights $\mathbf{W}^l$ and biases $\mathbf{b}^l$ are normalized to $\tilde{\mathbf{W}}^l \leftarrow \mathbf{W}^l \frac{\lambda^{l-1}}{\lambda^l}$ and $\tilde{\mathbf{b}}^l \leftarrow \mathbf{b}^l/\lambda^l$.

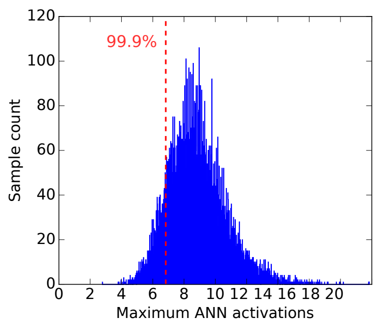
3.2.2 Robust mormalization

Although weight normalization avoids saturating firing rates in the SNN, it might result in very low firing rates, thereby increasing the latency until information reaches the higher layers. In the algorithm sketched above in Section 3.2.1, which we refer to as "max-norm", the normalization factor λ^l by which the weights and biases are scaled was set to the maximum ANN activation among all samples of the training set. This is a very conservative approach, which ensures that the SNN firing rates never exceed the maximum firing rate. The drawback is that this procedure is prone to be influenced by singular outlier samples that lead to very high activations, while for the majority of the remaining samples, the firing rates will remain considerably below saturation. Such outliers are not uncommon, as shown in Figure 1a, which plots the distribution of all non-zero activations in the first convolution layer for 16666 CIFAR10 samples (in log-scale). The maximum observed activation is more than three times higher than the 99.9th percentile. Figure 1b shows the distribution of the highest activations across the 16666 samples for all ANN units in the same layer, revealing a large variance across the dataset, and a peak that is far away from the absolute maximum. This explains why normalizing by the maximum can potentially perform poorly: For the vast majority of samples even the maximum activation of units within a layer will lie far below the chosen normalization scale, and thus there is insufficient firing in the SNN to drive higher layers and obtain accurate results.

As a more robust alternative we propose that instead of choosing the maximum activation, we can set λ to the p-th percentile of the total activity distribution. This discards extreme outliers, and increases SNN firing rates for a larger fraction of samples. The potential drawback is that a small percentage of neurons will saturate, so choosing the normalization scale involves a trade-off between saturation and insufficient firing. In the following, we refer to the percentile p as the "normalization scale", and note that the "max-norm" method is recovered as the special case $p = 100$. Typical values for p that perform well are in the range $[99.0, 99.999]$. In general, saturation of a small fraction of neurons seems to degrade network performance less than having too low spike rates. This method can be combined with using batch-normalization during ANN training (Ioffe and Szegedy, 2015), which standardizes the activations in each layer and therefore produces fewer extreme outliers.



(a)



(b)

Figure 1: (a) Distribution of all non-zero activations in the first convolution layer of a CNN, for 16666 CIFAR10 samples, in log-scale. The dashed line in both plots indicates the 99.9th percentile of all ReLU activations across the dataset, corresponding to a normalization scale $\lambda = 6.83$. This is more than three times less than the overall maximum of $\lambda_{max} = 23.16$. (b) Distribution of maximum ReLU activations for the same 16666 CIFAR10 samples. For most samples their maximum activation is far from λ_{max}.

3.3 Conversion of Batch-normalization layers

Batch-normalization (BN) (Ioffe and Szegedy, 2015) reduces internal covariate shift in ANNs and thereby speeds up the training process. BN introduces additional layers where affine transformations of inputs are performed in order to achieve zero-mean and unit variance. An input x is transformed into $\mathrm{BN}[x] = \frac{\gamma}{\sigma}(x - \mu) + \beta$, where mean μ, variance σ, and the two learned parameters β and γ are all obtained during training as described in (Ioffe and Szegedy, 2015). After training, these transformations can be integrated into the weight vectors, thereby preserving the effect of BN, but eliminating the computations. Specifically, we set $\tilde{W}^l_{ij} = \frac{\gamma^l_i}{\sigma^l_i} W^l_{ij}$ and $\tilde{b}^l_i = \frac{\gamma^l_i}{\sigma^l_i}\left(b^l_i - \mu^l_i\right) + \beta^l_i$. This makes it simple to convert BN layers into SNNs, because after transforming the weights of the preceding layer, no additional conversion for BN layers is necessary. Empirically we found loss-less conversion if BN parameters are integrated into other weights like this, the advantage lies purely in obtaining better ANNs using BN during training.

3.4 Analog input to first hidden layer

Because truly event-based benchmark datasets are rare (Hu et al., 2016), conventional frame-based image databases such as MNIST (LeCun et al., 1998) or CIFAR (Krizhevsky and Hinton, 2009) have been used to evaluate the accuracy of the SNN after conversion. Previous work has usually converted analog input activations, e.g. gray levels or RGB values into Poisson firing rates. But this introduces variability into the firing of the network and impairs its performance, without having any notable benefits. A simple alternative is to use analog input values in the very first hidden layer, and compute with spikes from there on (Zambrano and Bohte, 2016). Empirically we found this to be particularly effective in the low-activation regime of ANN units, where usually undersampling in spiking neurons poses a challenge for successful conversion.

3.5 Spiking softmax

Softmax is commonly used as the output of a deep ANN, because it results in normalized and strictly positive class likelihoods. Previous approaches for ANN-to-SNN conversion could not convert softmax layers, but simply predicted the class corresponding to the neuron that spiked most during the presentation of the stimulus. However, this approach fails when all neurons in the final layer receive negative inputs, and thus never spike.

Here we convert ANN softmax layers by using a mechanism proposed in (Nessler et al., 2009), where output spikes are triggered by an external Poisson generator with variable firing rate. The spiking neurons do not fire on their own but simply accumulate their inputs. When the external generator determines that a spike should be produced, a softmax competition according to the accumulated membrane potentials is performed.

3.6 Spiking max-pooling layers

Most successful ANNs use max-pooling to spatially down-sample feature maps, but this has not been used in SNNs because computing maxima with spiking neurons is non-trivial. Instead, simple average pooling had been used in (Cao et al., 2015; Diehl et al., 2015), which results in using weaker ANNs before conversion. Lateral inhibition, as suggested in (Cao et al., 2015), does not fulfill the job properly, because it only selects the winner, but not the actual maximum firing rate. Another suggestion by (Orchard et al., 2015) is to use a time-to-first-spike encoding, in which the first neuron to fire is considered the maximally firing one. Here we propose a simple mechanism for spiking max-pooling, in which output units contain gating functions, which only let spikes from the maximally firing neuron pass, while discarding spikes from other neurons. The gating function is controlled by computing estimates of the pre-synaptic firing rates, e.g. by computing an online or exponentially weighted average. In practice we found several methods to work well, but demonstrate only results using exponentially weighted averages of firing rates to control the gating function.

4 Results

There are two ways to improve the accuracy of the SNN via conversion: 1) training a better ANN before conversion, and 2) improving the conversion by eliminating approximation errors of the SNN. In the following we show the influence of both approaches.

4.1 Contribution of improved ANN architectures

The methods introduced in Section 3 allow conversion of CNNs that use biases, softmax, batch-normalization, and max-pooling layers, which all improve the accuracy of the ANN. This was quantified on the CIFAR10 benchmark (Krizhevsky and Hinton, 2009), using a CNN with 4 convolution layers (32 3x3 - 32 3x3 - 64 3x3 - 64 3x3), ReLU activations, batch-normalization, 2x2 max-pooling layers after the 2nd and 4th convolutions, followed by 2 fully connected layers (512 and 10 neurons) and a softmax output. This ANN achieved 87.86% accuracy. Constraining the biases to zero reduced the accuracy to 87.73%. Replacing max-pooling by average-pooling further decreased the accuracy to 87.69%. Eliminating the softmax and using only ReLUs in the output led to a big drop to 69.44%. With our new methods we can therefore start the conversion already with much better ANNs than was previously possible.

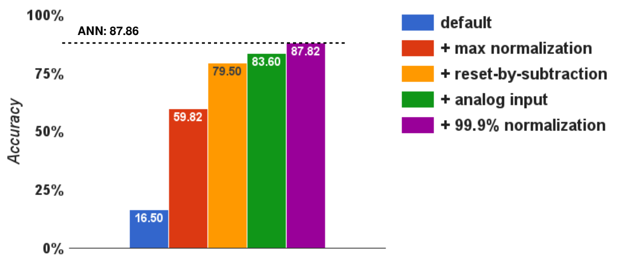
4.2 Contribution of improved SNN conversion methods

Figure 2a shows that in the case of CIFAR10 the conversion of the best ANN into an SNN using the default approach (i.e. no normalization, Poisson spike train input, reset-to-zero) fails, yielding an accuracy of 16.5%, barely above chance level. Adding weight normalization as suggested in (Diehl et al., 2015) (red bar) raises the accuracy to 59.82%, but this is still a big drop from the ANN result of 87.86%. Changing to the *reset-by-subtraction* mechanism from Section 2 leads to another 20% improvement (orange bar), and switching to analog inputs to the first hidden layer instead of Poisson spike trains results in an accuracy of 83.6% (green bar). Finally, using the 99.9th percentile of activations for robust weight normalization yields 87.62% accuracy, which is very close to the ANN performance and our best result for CIFAR10 with single SNNs. We can therefore conclude that all the proposed mechanisms for ANN training and ANN-to-SNN conversion contribute positively to the success of the method. The conversion into a SNN is nearly loss-less, and the results are very competitive for classification benchmarks using SNNs. These results were confirmed also on MNIST, where a 7-layer network with max-pooling achieved an accuracy of 99.44%, thereby improving previous state-of-the-art results for SNNs reported by (Diehl et al., 2015) and (Zambrano and Bohte, 2016).

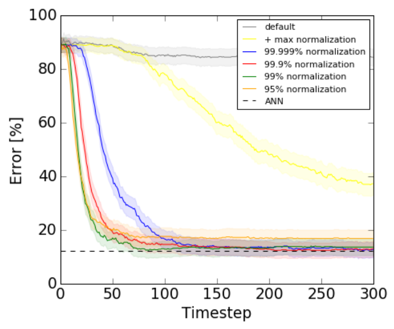
SNNs are known to exhibit a so-called accuracy-latency-tradeoff (Diehl et al., 2015; Neil et al., 2016), which means that the accuracy improves the longer the network is being simulated. In Figure 2b we show that the robust weight normalization factor can be tuned to ideally exploit this property. Empirically the best results were obtained with normalization factors corresponding to the 99th or 99.9th percentiles of activations. Both lead to accurate classifications quickly, and also converge to error rates very similar to those of the underlying ANN.

5 Discussion

By allowing a larger class of CNNs to be converted into SNNs, and by introducing a number of novel improved conversion techniques we could significantly improve the accuracy of our networks on both CIFAR10 and MNIST. Our best SNN result of 87.82% accuracy on CIFAR10 compares favorably to previous SNN results: (Cao et al., 2015) achieved 77.43% accuracy on CIFAR10, albeit with a smaller network and after cropping images to 24x24. With a similarly small network and cropped images, (Hunsberger and Eliasmith, 2015) achieve 82.95% accuracy. Both conversion methods lose less than 2% due to the conversion, but since their approach does not use weight normalization, the results shown in Figure 2a suggest that these methods would have problems on larger networks. Better SNN accuracies to date have only been reported by (Esser et al., 2016), where an accuracy of 89.32% was reported for a very large network optimized for 8 TrueNorth chips, and making use of ternary weights and multiple 1x1 network-in-network layers. A smaller network fitting on a single chip is reported to achieve 83.41%. In our own recent experiments with similar low-precision training schemes for SNNs we converted the BinaryConnect model by (Courbariaux and Bengio, 2016). Starting from an



(a)



(b)

Figure 2: (a) Influence of novel mechanisms for ANN-to-SNN conversion on the SNN accuracy for CIFAR10. The best ANN from Section 4.1 (87.86%) is converted into an SNN. Default mode (blue bar): SNN with Poisson inputs, reset-to-zero, and no weight normalization. Red bar: applying weight normalization as in (Diehl et al., 2015). For the next three bars we apply novel techniques as presented in Section 3. Shown is the accuracy after 300 time steps. (b) Accuracy-latency-tradeoff: SNNs give approximate results even when inputs are incomplete, and improve their accuracy with time. Tested on 400 CIFAR10 samples we find that the accuracy improves rapidly, and approaches the ANN level. The robust weight normalization factor can be tuned to achieve an ideal tradeoff between latency and final accuracy.

ANN with 91.94% accuracy, we achieved an accuracy of the SNN of 91.35% on CIFAR10, which is by far the best SNN result reported to date.

We know from ANNs that larger networks typically perform better than smaller networks, we thus fully expect to see a boost in SNN accuracy when applying the conversion techniques to even larger networks. In fact, the best ANNs to date achieve less than 5% error on CIFAR10 (Springenberg et al., 2014). Our goal here was to expand the toolkit for ANN-to-SNN conversion to the point where such large networks, using typical CNN mechanisms, can be converted into SNNs with only minimal loss of accuracy. That the drop-off is typically less than 1% is encouraging. We have also shown that the typical accuracy-latency tradeoff is still present, although our networks were not specifically trained to converge fast. Using the techniques proposed by (Neil et al., 2016) should yield accurate results even faster.

6 Conclusions

Deriving a first solid theory for ANN-to-SNN conversion has directly revealed mechanisms to improve the classification accuracy of the resulting SNN by a simple switch of reset mechanisms. This, together with novel tools to convert a large class of CNNs, covering most standard features of conventional CNNs, has helped achieving state-of-the-art SNN results, and almost loss-less ANN-to-SNN conversion. Future research will apply the methods to new datasets that require large networks, such as ImageNet. Another promising line of research is to investigate mechanisms that further reduce the number of spikes produced, eliminating redundancies when information about static inputs are sent. We have demonstrated that the accuracy gap between ANNs and SNNs can be almost completely closed.

Acknowledgments

We thank Jun Haeng Lee for helpful comments and discussions. This work has been supported by the Samsung Advanced Institute of Technology.

References

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66.

Courbariaux, M. and Bengio, Y. (2016). BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv*, page 9.

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

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., di Nolfo, C., Datta, P., Amir, A., Taba, B., Flickner, M. D., and Modha, D. S. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*.

Farabet, C., Paz, R., Pérez-Carrasco, J., Zamarreño, C., Linares-Barranco, A., LeCun, Y., Culurciello, E., Serrano-Gotarredona, T., and Linares-Barranco, B. (2012). Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel ConvNets for visual processing. *Frontiers in neuroscience*, 6:32.

Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. (2016). DVS benchmark datasets for object tracking, action recognition and object recognition. *Frontiers in Neuroscience*, 10:405.

Hunsberger, E. and Eliasmith, C. (2015). Spiking Deep Networks with LIF Neurons. *arXiv:1510.08829 [cs]*, pages 1–9.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, pages 1–11.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

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.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuromorphic Engineering*, 10.

Liu, S.-C., Delbruck, T., Indiveri, G., Whatley, A., and Douglas, R. (2014). *Event-Based Neuromorphic Systems*. John Wiley & Sons.

Neil, D. and Liu, S.-c. (2013). Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator. *IEEE transactions on very large scale integration systems*, 22(12):1–8.

Neil, D., Pfeiffer, M., and Liu, S.-C. (2016). Learning to be efficient: algorithms for training low-latency, low-compute deep spiking neural networks. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 293–298. ACM.

Nessler, B., Pfeiffer, M., and Maass, W. (2009). STDP enables spiking neurons to detect hidden causes of their inputs. In *Advances in neural information processing systems*, pages 1357–1365.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7:178.

Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., and Benosman, R. (2015). HFirst: A temporal approach to object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2028–2040.

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. application to feed forward convnets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719.

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T. (2014). Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Stromatias, E., Neil, D., Galluppi, F., Pfeiffer, M., Liu, S.-C., and Furber, S. (2015). Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on spinnaker. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Zambrano, D. and Bohte, S. M. (2016). Fast and efficient asynchronous neural computation with adapting spiking neural networks. *arXiv preprint arXiv:1609.02053*.