Minimizing Inference Time: Optimization Methods for Converted Deep Spiking Neural Networks

Etienne Mueller, Julius Hansjakob, Daniel Auge, Alois Knoll
Department of Informatics, Technical University of Munich, Munich, Germany
etienne.mueller@tum.de, knoll@in.tum.de

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533874

***Abstract*—Spiking neural networks offer the potential to drastically reduce energy consumption in edge devices. Unfortunately they are overshadowed by today's common analog neural networks, whose superior backpropagation-based learning algorithms frequently demonstrate superhuman performance on different tasks. The best accuracies in spiking networks are achieved by training analog networks and converting them. Still, during runtime many simulation time steps are needed until they converge. To improve the simulation time we evaluate two inference optimization algorithms and propose an additional method for error minimization. We assess them on Residual Networks of different sizes, up to ResNet101. The combination of all three is evaluated on a large scale with a RetinaNet on the COCO dataset. Our experiments show that all optimization algorithms combined can speed up the inference process by a factor of ten. Additionally, the accuracy loss between the original and the converted network is less than half a percent, which is the lowest on a complex dataset reported to date.**

***Index Terms*—spiking neural networks, conversion, object detection, residual networks, neuromorphic computing**

I. Introduction

Neural networks find their way into a growing number of applications with restricted power budgets. Their computational advantage has been demonstrated in many areas. Ranging from sensor data processing in edge devices to artificial intelligence (AI)-backed decision processes of autonomous vehicles, the networks have to evaluate large amounts of data and consume as little energy as possible. A further reduction of the consumed energy can result in longer battery life times of edge devices or an increased range of electrical vehicles.

To reduce the overall energy consumption of the networks, current research directions are twofold: developing new network architectures and optimizing the hardware running the network itself. A third direction, which gained traction in the recent past, is the utilization of biologically inspired neurons to realize the networks. Instead of continuous-valued activation functions, these neurons communicate via short all-or-nothing pulses and leverage time as the main information carrier. This has shown to provide a superior computational power compared to standard activation functions in analog neural network (ANN) [22]. Additionally, these spiking neural networks (SNN) promise an ultra-low powered hardware feasibility due to their simple integrate-and-fire computations.

Neuromorphic hardware is the silicon realization of SNNs. While still being in an early phase of research, there are multiple different chips available for research, e.g. Loihi by Intel [4], TrueNorth by IBM [23] and SpiNNaker [27]. In the future these and many other realizations will be available for real-world applications. Neuromorphic Hardware has the chance to revolutionize how neural networks are deployed today. By mimicking the neurons in the brain on a silicon chip they only consume energy when spikes are emitted between each other. The most optimistic estimates range anywhere between $10^1 - 10^5$ times less energy consumption compared to classical computing [29]. To that level is a long way to go, but today's research hardware already shows an energy reduction of 4x [1].

Although SNNs show promising advantages compared to ANNs, they are not widely used in applications yet. One reason behind that is the large controversy about learning algorithms for SNNs. Many supervised and unsupervised, often biologically inspired learning rules, like spike-timing-dependent plasticity [3], are still under active research. Backpropagation, the common learning algorithm in ANNs, is now slowly adopted to the field of SNNs. Because of the time dependency of spiking neurons it is not possible to use their derivative, so a pseudo derivative has to be used [25]. However, best-in-class results on popular benchmarks have been reached using methods which directly convert pre-trained ANNs into SNNs [38].

Accordingly, using conversion-based methods combines the best of both worlds: using the superior training algorithm in ANNs and porting the resulting weights to SNNs for the deployment in neuromorphic hardware. Early effort, however, showed a large loss after conversion, because the standard activation function back then, sigmoid, did not translate well to the firing frequency of a spiking neuron [28]. Since the use of rectified linear units (ReLU) [24] as activation function, things have changed. Because both the activation of ReLU as well as the firing rate of leaky integrate-and-fire (LIF) neuron increase linearly with their input, only a negligible conversion loss occurs after replacement [2]. In regards of performance, converted SNNs have been the most successful [38].

Still, there remains a gap in performance between the original and the converted network. Additionally, due to the temporal characteristics of SNNs, the optimal result during inference is reached after a certain convergence time. Since the spikes emitted by the different neurons have to propagate through the network, this inference time grows with the depth of the overall network.

We thank Infineon Technologies AG for supporting this research. The first two authors contributed equally to this work.

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

Accordingly, to apply conversion to large, state-of-the-art ANN, the time to converge towards the optimal output during inference needs to be reduced. In time constrained applications, this directly leads to an improved network performance if the full convergence can be reached during the given evaluation time.

In this work, we present a novel solution and evaluate two methods to decrease inference time. After giving a short overview over related work, we introduce three modifications for current conversion methods. Subsequently, we demonstrate their impact on the conversion accuracy and inference time on the small CIFAR10 benchmark dataset [16]. Finally, we show the joint result of all modifications on the Microsoft Common Objects in Context (COCO) dataset [20]. The resulting mean average precision (mAP) of 30.09 marks the best performing spiking object detection to our knowledge so far. Additionally, our approach reduces the accuracy loss between the original and the converted network to under half a percent, which undercuts previous work for large datasets.

II. Related Work

A. Object Detection

Object detection describes the task of localizing and classifying objects within an image. They can be categorized into two classes: single-step and two-step detectors. A good overview was created by Zhao et al. [43]. Two-step detectors consists of a region proposal step to extract areas with possible objects, and a subsequent classification step to evaluate the content. One-step detectors combine the regression and classification task into a single neural network and output the bounding boxes with corresponding class probability at the same time. Object detection cannot be evaluated by accuracy, since there are infinitely many true negatives. Instead the mean average precision is used.

Residual Networks. ResNets are a special architecture for deep convolutional neural network (CNN) proposed by He et al. [7]. Because of the good performance, ResNet-like architectures have become the defacto standard for CNN. Many architectures following the same design principle have been presented since then, the most mentionables being ResNeXt [41], DenseNet [10], Inception-v4 [35], MobileNetV3 [8] and EfficientNet [36]. He et al. proposed in their work ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152 with the number depicting the number of layers of the network. ResNets consists of multiple residual blocks with inherent shortcut connections. That enables the network to ignore layers which are not beneficial.

RetinaNet. RetinaNet is a one-step object detection architecture with a feature pyramid network (FPN) [18] and trained with focal loss [19], which significantly improves the performance . RetinaNet consists of four subnetworks: a residual network (ResNet) extracts features from the input image by generating feature maps, an FPN combines feature maps of different resolutions in order to have semantically rich feature maps of low and high resolution and two networks for classification and bounding box regression. Similar working one-step object detectors include SSD [21], YOLOv2 [30] and YOLOv3 [31] and EfficientDet [37].

B. Conversion

In earlier work many different conversion approaches have been showed. Rueckauer et al. [32] presented a theoretical basis which explained the underlying mechanism and different spiking implementations of ANN operators like max-pooling, batch normalization [12] and an improved version of the softmax mechanism by Nessler et al. [26]. This work marked the foundation for subsequent research, for what reason we refer to these collective methods as the *basic conversion*.

For the conversion from ANN to SNN, the spike rate $r(t)$ can be calculated from the the ReLU activation a of the original network, as showed in the supplementary material by Rueckauer et al. [32]:

$$r(t) = ar_{max} - \frac{V(t) - V(0)}{tV_{thr}} \tag{1}$$

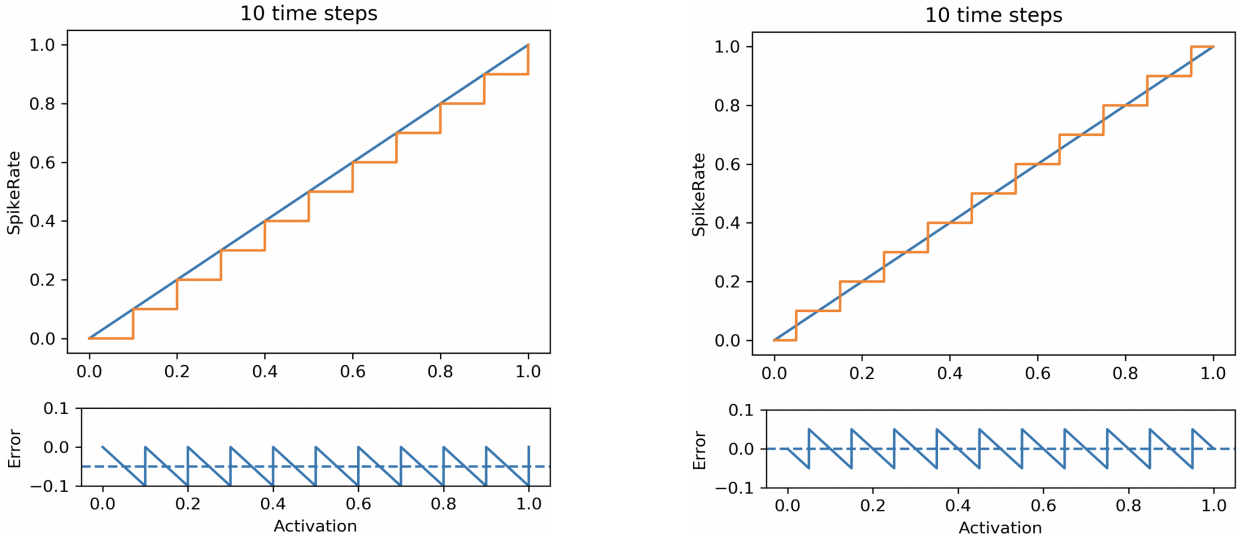
with $V(t)$ being the membrane potential and V_{thr} the firing threshold. As pointed out by Diehl et al. [5], the spiking neuron, in contrast to ReLU, has an upper bound determined by the maximum spike rate r_{max}. Therefore the weights of the ANN have to be normalized to fit into the spike frequency range. Because this approach ensures that the maximum firing rates are never exceeded, single outliers can drastically decrease weights and thus increase inference time of the network. Rueckauer et al. [32] proposed to discard extreme outliers by using *robust normalization* where only the p^{th} percentile of the total activity distribution per layer is used. The best results could be achieved with values for p in the range $[99.0, 99.999]$. Kim et al. [14] further improved this method and introduced *channel-wise normalization*, which led to an at least 2.3x increased inference speed on object detection dataset by normalizing each channel individually.

Another approach for spiking object detection where introduced by Hu et al. [9], who showed conversion methods for ResNets. Xiao et al. [40] and Kerapdiseh et al. [13] showed biologically plausible learning algorithms for object recognition. Recently, Wu et al. [39] introduced Progressive Tandem Learning, a layer-wise learning framework with an adaptive training scheduler for rapid pattern recognition.

III. Method

During inference of a spiking network the input signal takes multiple time steps to pass through the network. The initially created output mainly results from highly activated neurons and gets influenced by more and more neurons the longer the simulation runs. During this *transient phase* the accuracy starts significantly reduced and converges to its maximum over time. Depending on number of layers and neurons, many time steps need to be calculated before an accurate prediction can be made from the resulting output.

Our goal is to reduce the necessary inference time to a minimum. Therefore we evaluate the optimization methods



(a) Biased quantization

(b) Unbiased quantization

Fig. 1: Overview of the biased and unbiased quantization methods in spiking neural networks

voltage clamping by Rueckauer et al. [32] and *channel-wise normalization* by Kim et al. [14]. Furthermore we propose *unbiased quantization*, a method to reduce the error in converted networks.

Voltage Clamping. To improve performance during the transient phase Rueckauer et al. [32] proposed to clamp the membrane potential for each layer l in the network for the first N time steps by $N(l) = d \cdot l$, with d being the delay between lifting the clamp from consecutive layers. With this additional neurons can accumulate spikes from previous layers and therefore include signals from neurons with slower firing frequency which increases the overall accuracy in the output signal. The authors found a clamping delay of $d = 10$ for Inception-V3 sufficient, it did however not significantly change the accuracy on VGG16. d introduces another hyperparameter which will be further evaluated in this work.

Channel-Wise Normalization. The basic conversion method used layer-wise normalization, where the channels within a feature map are collectively normalized. Kim et al. [14] remarked that there is a large discrepancy between the maximum activation of the channels within a feature map of a convolutional neural network. Therefore, they propose to normalize each channel individually. The weights and biases are normalized with

$$W^l_{i,j} \leftarrow W^l_{i,j} \frac{\lambda^{l-1}_i}{\lambda^l_j} \quad \text{and} \quad b^l_j \leftarrow \frac{b^l_j}{\lambda^l_j} \tag{2}$$

with λ^k_c being the p^{th} percentile of the c^{th} channel in the k^{th} layer. The authors utilize the 99.9^{th} percentile for the normalization, but this method also introduces another hyperparameter which needs additional adjustment.

Unbiased Quantization. With rate encoding, the continuous activation of an analog neuron is quantized. The basic conversion rounds the activation down to the nearest quantization step. As a result, if the activations are uniformly distributed, the quantization error is biased by $\frac{1}{2t}$ (see figure 1a). Yousefzadeh et al. [42] proposed a hysteresis quantization of the ReLU activation function before conversion, which results in evening out the error. We suggest to initialize the membrane potential of the SNNs neurons with $\frac{1}{2}V_{thr}$ which will unbias the quantization error. Accordingly, the error is on average zero and the spike rate uniformly under- and overestimates the corresponding activation of the ANN (see figure 1b).

The quantization error $e^1_i(t)$ in the first hidden layer can be calculated with the spike rate $r^1_i(t)$ (see equation 1) and the activation a^1_i of the corresponding ANN neuron with an initial potential $V^1_i(0) = 0$:

$$\begin{aligned} e^1_i(t) &= |a^1_i - r^1_i(t)| \\ &= \left| a^1_i - \left(a^1_i - \frac{V^1_i(t)}{tV_{thr}} \right) \right| \\ &= \left| \frac{V^1_i(t)}{tV_{thr}} \right| \end{aligned} \tag{3}$$

With the activation $a^1_i > 0$ it is reasonable to assume that $0 < V^1_i(t) < V_{thr}$. Thus the quantization error has an upper limit of $e^1_i(t) < \frac{1}{t}$. If the neurons membrane potential is instead initialized with $\frac{1}{2}V_{thr}$ the bound is halved:

$$\begin{aligned} e^1_i(t) &= \left| a^1_i - \left(a^1_i - \frac{V^1_i(t) - \frac{1}{2}V_{thr}}{tV_{thr}} \right) \right| \\ &= \left| \frac{V^1_i - \frac{1}{2}V_{thr}}{tV_{thr}} \right| < \frac{1}{2t} \end{aligned} \tag{4}$$

The same applies to deeper layers. The spike rate $r^l_i(t)$ of a layer $l > 1$ can be calculated as follows:

$$r^l_i(t) = \sum_{j=1}^{M_{l-1}} W^l_{i,j} r^{l-1}_j(t) + r_{max} b^l_i - \frac{V^l_i(t) - V^l_i(0)}{tV_{thr}} \tag{5}$$

With $r_{max} = 1$, the quantization error results as follows:

$$\begin{aligned} e_i^l(t) = \left|a_i^l - r_i^l(t)\right| = & \left| \left(\sum_{j=1}^{M_{l-1}} W_{i,j}^l a_j^{l-1} + b_i^l \right) \right. \\ & \left. - \left(\sum_{j=1}^{M_{l-1}} W_{i,j}^l r_j^{l-1}(t) + b_i^l - \frac{V_i^l(t) - V_i^l(0)}{tV_{thr}} \right) \right| \\ = & \left| \sum_{j=1}^{M_{l-1}} W_{i,j}^l e_j^{l-1}(t) + \frac{V_i^l(t) - V_i^l(0)}{tV_{thr}} \right| \end{aligned} \tag{6}$$

The quantization error for $V_i^l(0) = 0$ is then limited by:

$$\begin{aligned} e_i^l(t) &= \left| \sum_{j=1}^{M_{l-1}} W_{i,j}^l e_j^{l-1}(t) + \frac{V_i^l(t)}{tV_{thr}} \right| \\ &< \left| \sum_{j=1}^{M_{l-1}} W_{i,j}^l e_j^{l-1}(t) + \frac{1}{t} \right| \end{aligned} \tag{7}$$

If the neurons membrane potential is initialized with $\frac{1}{2}V_{thr}$, the maximal error the neuron adds due to the quantization is halved.

$$\begin{aligned} e_i^l(t) &= \left| \sum_{j=1}^{M_{l-1}} W_{i,j}^l e_j^{l-1}(t) + \frac{V_i^l(t) - \frac{1}{2}V_{thr}}{tV_{thr}} \right| \\ &< \left| \sum_{j=1}^{M_{l-1}} W_{i,j}^l e_j^{l-1}(t) + \frac{1}{2t} \right| \end{aligned} \tag{8}$$

Initializing SNNs with this method centers the error and thus results in a more accurate prediction in less simulation time. In addition it does not introduce any further hyperparameters.

IV. Experiments

For a small scale evaluation of each of the three different optimization methods and their joint combination, we train four ResNets (ResNet18 to ResNet101) on CIFAR10 [16]. This classification dataset contains 60,000 images of size 32×32×3 pixels, categorized into 10 classes. The training set consists of 50,000, and the test set of 10,000 images. Since CIFAR10 does not contain a validation dataset, we use the last 5,000 images of the training set for this purpose. Therefore, the training set is reduced to its first 45,000 images.

In order to get an unbiased result, we separately evaluate unbiased normalization, voltage clamping as well as channel-wise normalization on the validation set. Then the determined best performing hyperparameters are collectively assessed on the test set.

To evaluate the effectiveness of the methods on a larger scale, we convert a RetinaNet with a ResNet18 as backbone to an SNN and assess its performance on the 2017 COCO dataset by Microsoft [20]. COCO is a dataset for object detection, segmentation, and captioning. It contains roughly 118,000 training, 5,000 validation, and 41,000 test images with three RGB channels, totaling approximately 164,000 images. The objects are categorized in 80 classes. The images have various resolutions ranging from 72×51 to 640×640 pixels and varying aspect ratios of up to 6:1. Due to the network's high computational complexity, the performance is only assessed on the 5,000 validation images. All optimization methods are jointly applied in this last experiment.

For simplicity we use potential encoding for the output layer [32], where the softmax is calculated on the membrane potential of the last layer. All SNNs are evaluated for 1,000 time steps.

A. Evaluation of the Optimization Methods

First we compare the unbiased quantization, the channel-wise normalization and the voltage clamping with the hyperparameter value $d = [1, 2, 3]$. In this step we keep the normalization to the 99.9^{th} percentile, as this was proposed in the work by Kim et al. [14]. Subsequently, we examine the normalization parameters and check whether the 99.9^{th} percentile is optimal in this application, too. Because Rueckauer et al. [32] reported that percentiles in the range $[99, 99.999]$ perform the best for the layer-wise normalization, we evaluate the values $[99, 99.9, 99.99, 99.999, 100]$ for the channel-wise normalization collectively with the other two methods in the best configuration found in the previous experiment. Additionally to the simulation of the converted SNNs, the ReLU1 performance of the normalized ANNs are computed. For this, all ReLU activation functions are replaced with the ReLU1 function. Similar to the SNN's normalized spike frequency response, this clips all activations between 0 and 1. The result gives insight of how much the accuracy drop can be attributed to the clipping and how much to the quantization errors of the SNN. Additionally, the ReLU1 performance should give a theoretical, approximate upper bound on the SNN performance. After assessing the previous experiments on the validation set of CIFAR10, we evaluate the best determined configuration on the test dataset to get an unbiased result.

Training of the ResNets. Because of the small size of only 32×32 pixels, the ResNets have to be slightly adapted. The first convolution is changed to a 3x3 convolution with a stride of 1 and the max pooling layer is removed. Further, the number of channels is halved for all convolutions. The networks are trained for $50+100n$ epochs with $n = [1, 2, 3, 4]$ for ResNet18, ResNet34, ResNet50, and ResNet101. We use stochastic gradient descent (SGD) with a batch size of 128, an initial learning rate of 0.1, momentum of 0.9 and a weight decay of 0.0005 during training. The learning rate is divided by 10 after $50n$ and $25+75n$ epochs. The images are normalized with the per-channel mean and standard deviation such that each channel has a mean of 0 and a standard deviation of 1. We augment the data as described be He et al. [7] and Lee et al. [17] by zero-padding the images with 4 pixels on each side and then randomly cropping 32x32 pixels. Additionally each image is flipped horizontally with a probability of 50%.

Conversion of the ResNets. Since the converted activations of the ReLU after the element wise addition of the residual block

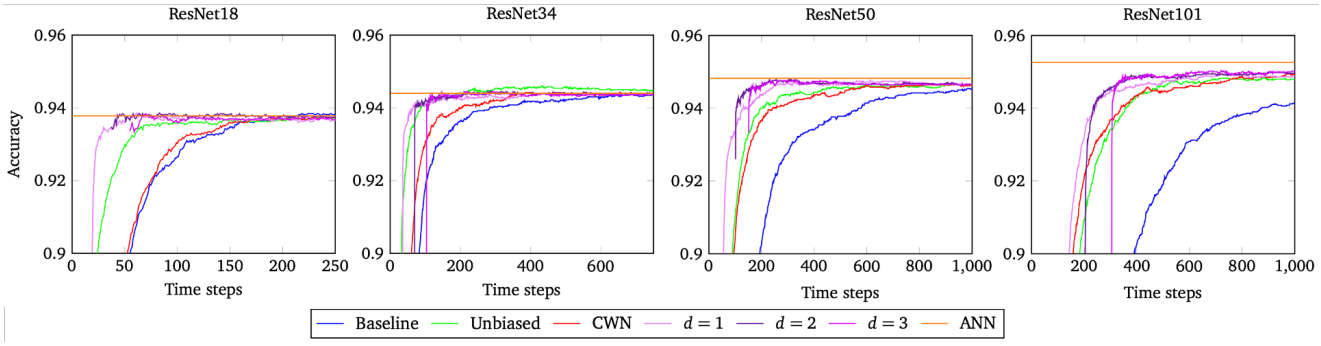


Fig. 2: Inference of the different optimization methods for ResNets of different sizes on the CIFAR10 dataset.

	ResNet18		ResNet34		ResNet50		ResNet101	
Percentile	SNN	ReLU1	SNN	ReLU1	SNN	ReLU1	SNN	ReLU1
99	92.24%	92.24%	92.42%	92.42%	92.84%	92.86%	92.66%	92.66%
99.9	93.71%	93.70%	94.26%	94.24%	94.71%	94.70%	95.01%	95.04%
99.99	93.85%	93.82%	94.40%	94.38%	94.78%	94.80%	95.27%	95.24%
99.999	93.80%	93.76%	94.40%	94.44%	94.74%	94.80%	95.28%	95.24%
100	93.87%	93.78%	94.35%	94.44%	94.78%	94.82%	95.25%	95.26%
ANN	93.78		94.40		94.82		95.26	

TABLE I: Accuracy for the channel-wise normalization combined with unbiased quantization and voltage clamping for different percentiles as well as their corresponding ReLU1 activation. For the spiking networks the average accuracy of the last 100 time steps is given.

should not exceed r_{max}, the residual blocks of the ResNets need to be normalized in a specific manner. Both, the main paths and the shortcuts, are normalized with their activations as described by Hu et al. [9].

B. Large Scale Evaluation

As presented in IV-A a reasonable percentile for the channel-wise normalization can be picked before conversion by computing the mAP of the ReLU1 with different values.

Training of RetinaNets. Our analog RetinaNet is trained slightly differently compared to the original work by Lin et al. [19]. Instead of SGD we use Adam [15] as optimizer with a batch size of 14, an initial learning rate of 0.0001, a weight decay 0.00001 and the hyperparameters set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The network is trained for a total of 90,000 iterations, after 60,000 and 80,000 iterations the learning rate is divided by 10. In contrast to the original paper the weight introduced by the focal loss is considered as a constant in the derivation step.

During training the shorter size of an image is randomly rescaled somewhere between 640 to 896 pixels. The batch of 14 images must all have the same resolution, therefore, all images are randomly zero-padded appropriately. Additionally, the images are horizontally flipped with a probability of 50%.

Conversion of RetinaNet. For the conversion of the FPN, additional considerations need to be made. The weights of the FPN have to be adapted due to the normalization in the backbone. The four convolutions that receive the feature maps C_3, C_4, or C_5 as input need to denormalize them. The activation of the 3x3 convolution after C_5 has to be normalized for the subsequent ReLU. That activation then gets denormalized again by the following 3x3 convolution. After denormalizing P_6 all pyramid feature maps are consistent and continue to the subsequent regression and classification subnetwork.

Because convolution and nearest neighbor upscaling are both linear functions, any combination of these two functions will result in a linear function. Therefore, all computations of the Feature Pyramid Network, except ReLU, can be condensed into their subsequent weights.

C. Results

Figure 2 shows the comparison of our proposed unbiased normalization, the channel-wise normalization with the 99.9^{th} percentile and the voltage clamping with $d = [1, 2, 3]$ to a baseline SNN. These three optimization methods all reach roughly the same accuracy and converge more quickly than the baseline spiking ResNet. Because of the size of the neural networks and the simplicity of the CIFAR10 dataset, we achieve lossless conversion for ResNet18 and ResNet34. The baseline spiking ResNet50 and ResNet101 do not converge within 1,000 time. However, at the end of the simulation their classification accuracy is still increasing. It is to assume that they would also converge to the accuracy of the original ANN.

Results of individual methods. Voltage clamping shows the biggest improvement of the tested methods on all the ResNets in term of inference time even though the network needs the number of layers multiplied by d time steps before it produces the first output. The different values for the hyperparameter d make no large difference. For $d = 1$ the network needs slightly more time steps to converge, whereas $d = 3$ needs three

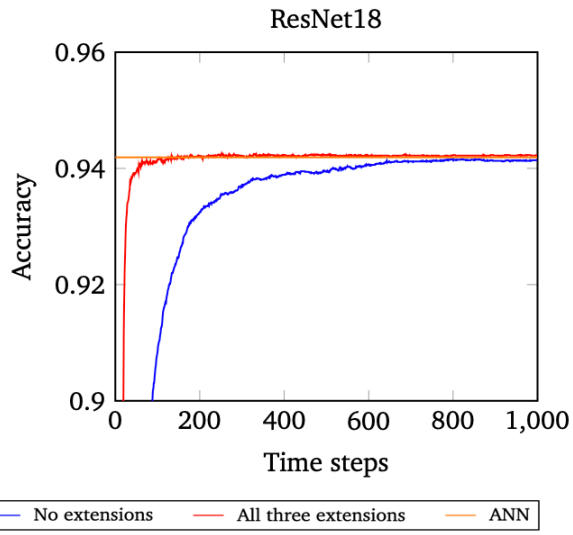


Fig. 3: Accuracy of the three optimization methods on the CIFAR10 dataset for a ResNet18.

ResNet	18	34	50	101
ANN	94.19%	94.67%	95.11%	94.92%
Basic	94.13%	94.41%	94.59%	93.45%
Extended	94.22%	94.61%	95.12%	94.95%

TABLE II: Resulting accuracies for the original ANN, the basic and the extended conversion. For the spiking networks the average accuracy of the last 100 time steps is given.

times longer to produce its first output, but then converges at a higher rate. For the following experiments we use $d = 1$ since the difference in accuracy can be neglected and higher values would delay the output generation for too long.

The second best improvement could be obtained with the unbiased quantization. It alone can more than halve the inference time compared to the baseline SNN. Although it converges slower than the voltage clamping method, it does not introduce another hyperparameter. Thus it can be implemented with no expense.

The channel-wise normalization makes the least difference, especially for smaller networks. Although it converges to the same accuracy as the other methods, it is just barely faster than the baseline ResNet18. For deeper networks the bigger difference is made by this more fine grained normalization and it performs more comparable to the other methods. For the ResNet101 it results in an even slightly faster inference than the unbiased quantization.

Since all optimization methods benefit the inference time, we further investigate the hyperparameter p for the channel-wise normalization in combination with the other two approaches. The results of the final accuracies for the SNNs, calculated by the average accuracy of their last 100 time steps, as well as the corresponding ReLU1 activation can be found in Table I.

The 99^{th} percentile performed the worst for both the SNN and the ANN, converging to a lower accuracy compared to the other methods of 1.5 to 2.5 percentage points. The 99.9^{th} percentile also converges to a slightly lower accuracy compared to the networks normalized with a higher percentile, which is most noticeable for the ResNet101. The 99.99^{th}, 99.999^{th} and the 100^{th} percentile normalization converge for all ResNets to the same accuracy as the ANN with just little noise in difference. The 100^{th} percentile however needs nearly twice as long to fully converge. Hence if inference speed is relevant, tuning the hyperparameter should be considered.

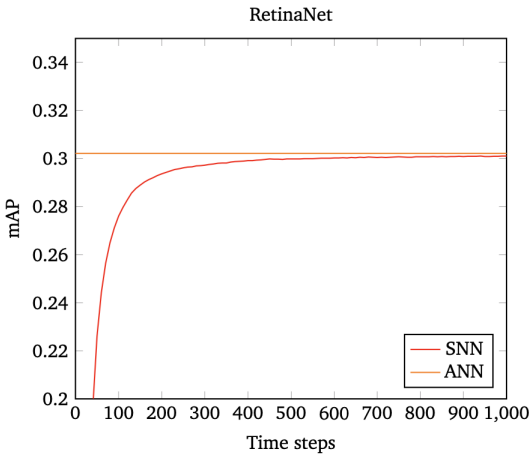


Fig. 4: mAP of the three optimization methods on the COCO dataset for a converted RetinaNet with a ResNet18 as backbone, as well as the corresponding ANNs mAP.

The ReLU1 accuracy is very close to the SNN's performance. Consequently it can be used as a predictor to evaluate different percentiles before conversion. It also suggest that the primary reason for accuracy difference is the clipping of activations and not the quantization errors of the SNN.

Extended Conversion. For the combination of the three optimization methods, we evaluate the best configuration on the CIFAR10 test set. That is unbiased quantization, channel-wise normalization with the 99.99^{th} percentile and voltage clamping with $d = 1$. We refer to this as *extended conversion* and compare it to the basic conversion.

Our approach performs significantly better compared to the basic conversion. We achieve approximately a tenfold speed increase on ResNet18 (see figure 3) with only 56 time steps until it crosses the 94% bar whereas the basic conversion needs roughly 600 steps to do so. The basic spiking ResNet101 does not even fully converge in the 1,000 time steps of the simulation, whereas the extended conversion still needs less than 150 time steps.

For ResNet34, and ResNet50 a noticeable accuracy loss can be found from the basic conversion to the original ANN. Using the extended conversion the converted networks reach the same accuracy as the ANN. Thus, the conversion for simple datasets like CIFAR10 is lossless. The resulting accuracies at the end of the simulation can be found in table II.

For the large scale evaluation of the extended conversion

	Network	Dataset	ANN	SNN	Loss
Diehl et al. (2015) [5]	3-layer CNN	MNIST	99.14%	99.10%	<0.1%
Stromatias et al. (2017) [34]	2-layer CNN	MNIST	98.30%	98.32%	<0.1%
Hu et al. (2018) [9]	ResNet8	MNIST	99.59%	99.59%	<0.1%
Yousefzadeh et al. (2019) [42]	4-layer CNN	MNIST	99.21%	99.19%	<0.1%
Cao et al. (2014) [2]	3-layer CNN	CIFAR10	79.12 %	77.43%	2.14%
Hunsberger, Eliasmith (2015) [11]	5-layer CNN	CIFAR10	85.97%	83.54%	2.43%
Rueckauer et al. (2017) [32]	9-layer CNN	CIFAR10	91.91%	90.85%	1.06%
Hu et al. (2018) [9]	ResNet44	CIFAR10	92.85%	92.37%	0.52%
Sengupta et al. (2019) [33]	ResNet34	CIFAR10	89.10%	87.46%	1.84%
Han et al. (2019) [6]	VGG16	CIFAR10	93.63%	93.63%	<0.1%
This work	**ResNet18**	**CIFAR10**	**94.19%**	**94.22%**	**<0.1%**
This work	**ResNet34**	**CIFAR10**	**94.67%**	**94.61%**	**<0.1%**
This work	**ResNet50**	**CIFAR10**	**95.11%**	**95.12%**	**<0.1%**
This work	**ResNet101**	**CIFAR10**	**94.92%**	**94.95%**	**<0.1%**
Hu et al. (2018) [9]	ResNet44	CIFAR100	70.18%	68.56%	1.62%
Han et al. (2019) [6]	VGG16	CIFAR100	71.22%	70.93%	0.29%
Rueckauer et al. (2017) [32]	VGG16	ImageNet	63.9% (84.9%)	49.6% (81.6%)	22.4% (3.8%)
Rueckauer et al. (2017) [32]	Inception-V3	ImageNet	76.1% (93.0%)	74.6% (92.0%)	2.0% (1.0%)
Sengupta et al. (2019) [33]	ResNet34	ImageNet	80.69% (89.69%)	65.47% (86.33%)	18.18% (3.75%)
Kim et al. (2019) [14]	YOLO	PascalVOC	53.01%	51.83%	2.23%
Kim et al. (2019) [14]	YOLO	COCO	26.24 mAP	25.66 mAP	2.21%
This work	**RetinaNet**	**COCO**	**30.21 mAP**	**30.09 mAP**	**0.40%**

TABLE III: Overview of Conversion Losses of Different Network Architectures and Datasets.

with a RetinaNet we computed the ReLU1 mAP for different percentiles to determine the hyperparameter of the channel-wise normalization. The 99.999^{th} performed the best with an mAP of 30.16, the 99.99^{th} came close with 29.68 and the 99.9^{th} performed the worst with a mAP of only 26.75. This demonstrates that a general value cannot be recommended and needs to be chosen for every application individually. The ReLU1 performance can be very well used for assessing an appropriate value.

In the experiment with 1,000 simulation time steps, the original ANN reaches an mAP of 30.21 and the converted SNN 30.09 (see figure 4). This corresponds to an absolute difference of just 0.12 or 0.40%.

V. Discussion

In this work we introduced unbiased quantization, a simple method to reduce the quantization error caused by the conversion of continuous activation functions to spike rates. We showed that this method greatly improves the inference time of converted SNNs. Additionally, we analyzed our method jointly with other recent modifications, namely channel-wise normalization and voltage clamping, for the conversion of ANNs to SNNs.

With optimized hyperparameters, the converted SNNs converge approximately ten times faster than networks using the basic conversion method on the CIFAR10 dataset. Additionally, the the conversion loss is reduced to a neglectable minimum. As shown in table III, the lossless conversion has only been demonstrated on the much simpler MNIST dataset so far.

A spiking RetinaNet using the extended conversion achieved an mAP of 30.09 with a hardly noticeable loss of 0.4% compared to the original ANN on the COCO dataset. Additionally to the higher mAP and lower conversion loss, our converted network converges much faster compared to Spiking-YOLO by Kim et al. [14]. In our simulation of 1,000 time steps, the spiking RetinaNet converges after 600 time steps whereas Spiking-YOLO was simulated for 8,000 time steps and needed roughly 3,000 time steps to reach convergence. Because different ANN architectures were evaluated, the overal mAP can not be directly compared. Though, the conversion loss introduced by the extended conversion is considerably lower.

In summary, the three methods from the extended conversion should be implemented to achieve the best results. For small networks, however, the channel-wise normalization can be omitted. Though, its usage adds an additional hyperparameter, the normalization percentile, which needs careful consideration to obtain the best performing result. Possible values of the hyperparameter can be examined by using the ReLU1 activation before conversion at lower expense. The voltage clamping is very important whenever a fast execution is desired. A small value for the hyperparameter d should then be chosen. Because of the simplicity of implementation and effectiveness on all spiking networks, we recommend unrestricted use of the unbiased quantization in SNNs.

References

[1] Peter Blouw and Chris Eliasmith. Event-Driven Signal Processing with Neuromorphic Computing Systems. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8534–8538, Barcelona, Spain, May 2020. IEEE.

[2] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *International Journal of Computer Vision*, 113(1):54–66, November 2014.

[3] Natalia Caporale and Yang Dan. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, 31(1):25–46, 2008.

[4] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A Neuromorphic Manycore

Processor with On-Chip Learning. *IEEE Micro*, 38(1):82–99, January 2018.

[5] Peter U. Diehl, D. Neil, J. Binas, Matthew Cook, S. Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2015.

[6] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13558–13567, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[9] Yangfan Hu, Huajin Tang, Yueming Wang, and Gang Pan. Spiking Deep Residual Network. *arXiv:1805.01352 [cs]*, April 2018.

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[11] Eric Hunsberger and Chris Eliasmith. Spiking Deep Networks with LIF Neurons. *arXiv:1510.08829 [cs]*, October 2015.

[12] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, March 2015.

[13] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, March 2018.

[14] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11270–11277, November 2019.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

[16] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. M.Sc. Thesis, University of Toronto, Department of Computer Science, 2009.

[17] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. *Proceedings of the AISTATS*, 2015.

[18] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*, February 2015.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 21–37, Cham, 2016. Springer International Publishing.

[22] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, December 1997.

[23] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, an Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface — Science. *Science*, 345(6197), August 2014.

[24] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, page 8, 2010.

[25] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks. *arXiv:1901.09948 [cs, q-bio]*, January 2019.

[26] Bernhard Nessler, Michael Pfeiffer, and Wolfgang Maass. STDP enables spiking neurons to detect hidden causes of their inputs. *Advances in Neural Information Processing Systems 22*, pages 1357–1365, 2009.

[27] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber. 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, August 2013.

[28] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begoña Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. 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, November 2013.

[29] Chi-Sang Poon and Kuan Zhou. Neuromorphic Silicon Neurons and Large-Scale Neural Networks: Challenges and Opportunities. *Frontiers in Neuroscience*, 5, 2011.

[30] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[31] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*, April 2018.

[32] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Frontiers in Neuroscience*, 11, 2017.

[33] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in Neuroscience*, 13, March 2019.

[34] Evangelos Stromatias, Miguel Soto, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. An Event-Driven Classifier for Spiking Neural Networks Fed with Synthetic or Dynamic Vision Sensor Data. *Frontiers in Neuroscience*, 11, 2017.

[35] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*, August 2016.

[36] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv:1905.11946 [cs, stat]*, November 2019.

[37] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020.

[38] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep Learning in Spiking Neural Networks. *Neural Networks*, 111:47–63, March 2019.

[39] Jibin Wu, Chenglin Xu, Daquan Zhou, Haizhou Li, and Kay Chen Tan. Progressive Tandem Learning for Pattern Recognition with Deep Spiking Neural Networks. *arXiv:2007.01204 [cs]*, July 2020.

[40] Will Xiao, Honglin Chen, Qianli Liao, and Tomaso Poggio. Biologically-plausible learning algorithms can scale to large datasets. *arXiv:1811.03567 [cs, stat]*, December 2018.

[41] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[42] Amirreza Yousefzadeh, Sahar Hosseini, Priscila Holanda, Sam Leroux, Thilo Werner, Teresa Serrano-Gotarredona, Bernabe Linares Barranco, Bart Dhoedt, and Pieter Simoens. Conversion of Synchronous Artificial Neural Network to Asynchronous Spiking Neural Network using sigma-delta quantization. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 81–85, Hsinchu, Taiwan, March 2019. IEEE.

[43] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, November 2019.