LOW PRECISION LOCAL LEARNING FOR HARDWARE-FRIENDLY NEUROMORPHIC VISUAL RECOGNITION

Jyotibdha Acharya, Laxmi R Iyer, and Wenyu Jiang

Institute of Infocomm Research, Singapore
{acharyaj, laxmi_r_iyer, wjiang} @i2r.a-star.edu.sg

ABSTRACT

Quantization is an important approach in making hardware-friendly implementation. However, while various quantization techniques have been extensively explored in deep learning for reducing the memory and computational footprint of the models, similar investigations are few in neuromorphic computing, which is supposed to demonstrate high power and memory efficiency over its more traditional counterpart. In this work, we explore quantization-aware-training (QAT) for SNNs as well as fully quantized transfer-learning using the DECOLLE learning algorithm as the basis system, whose local loss based learning is bio-plausible, avoids complex back-propagation-through-time and potentially hardware-friendly. We also evaluate different rounding functions, and analyze their effects on learning. We validate our results on two datasets, DVS-Gestures, and N-MNIST, where we reach within 0.3% difference from full precision accuracy for both datasets using only 3-bit weights with a convolutional neural network. We are currently exploring other datasets to understand the generalizability of the explored quantization schemes.

Index Terms— Spiking Neural Network (SNN), quantization, quantization-aware-training, local learning, transfer learning

1. INTRODUCTION

Spiking Neural networks are the third generation of neural networks, where neurons communicate through spikes, like in biological neurons. In contrast, traditional artificial neural networks (ANNs) compute using real numbers, which are typically implemented as floating point numbers. Therefore, expensive multiplications, which are a part of ANNs, are avoided in SNNs and this facilitates high power and memory efficiency. Such advantages can be further enhanced by quantization techniques, which make the system implementation hardware-friendly. This can come in two aspects: in the case where hardware implementation is pure digital, quantization will use fewer on-chip storage resources (such as SRAM) to store synaptic weights and may require a smaller arithmetic unit for computation of weighted sums; in the case where hardware implementation utilizes memristive and analog computing [1], proper quantization can facilitate storage of synaptic weights on memristor devices with limited precision, as well as limiting the impact of such limited precision on accuracy of the underlying algorithm.

In this work, we train a convolutional network with quantization-aware-training (QAT) using the DECOLLE local learning rules [2]. Moreover, we use online quantized retraining to add new classes to the model, similar to [3].

The primary contributions of this work are as follows:

- We use quantization-aware-training to train an SNN with local learning rule that can achieve near full-precision accuracy using only 3-bit weights.
- We explore the effects of different quantization functions on the performance of weight quantized network.
- We propose a transfer learning based methodology that can be used to fine-tune the quantized network to learn new classes through hardware-friendly online training.

2. RELATED WORK

The first SNNs were trained on Spike timing Dependent Plasticity (STDP), an unsupervised biological synaptic learning rule (E.g. [4], [5], [6]). While STDP is simpler to train and obviates the need for labelling, its accuracy on real world problems is relatively low. Therefore, variants of back-propagation based training algorithms were then developed for SNNs [7, 8] to improve accuracy.

Due to spikes' non-differentiable property and temporal credit assignment issue, classic back-propagation had to be modified using methods known as surrogate gradient (SG) as well as back-propagation-through-time (BPTT). SG based BPTT has shown state-of-the-art results in recent years for SNN training [7, 8]. Back-propagation in general involves global propagation of errors, and in addition, BPTT requires the network to store results for all intermediate timesteps to calculate the gradients. Therefore, it requires significant

This research was supported by Programmatic grant no. A1687b0033 from the Singapore government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain).

memory overhead and is not suitable for implementation on neuromorphic hardware [3]. In contrast, surrogate gradient based local learning rules such as DECOLLE [2], present a better trade-off where it achieves close to state of the art accuracy while avoiding computational, communication and memory overhead of BPTT.

One of the most prominent and well-researched solutions to the resource bottleneck of deep learning methods is weight quantization. A variety of methods, ranging from uniform quantization, symmetric/asymmetric quantization to more advanced ones such as knowledge distillation, have been developed over the years that enabled ANNs to achieve close to full accuracy even with binary weights in some cases [9]. However, surprisingly few works have been done on quantization of SNNs. In Wang et al. [10], authors have used conversion based methods to train and quantize ANNs and then convert them to SNNs while compromising on classification accuracy. STDP based methods have also been used to train binary weight kernels [11], but the methods have not been proven to work for event-based datasets. Wu et al. [12] has explored quantization in progressive (i.e. layer-wise) tandem learning for MNIST and CIFAR-10.

The issue of large data requirement and long training times for DL is often circumvented by using transfer learning, where the model is initially trained on a larger dataset and later retrained/fine-tuned for a context specific application using a smaller dataset [13]. While transfer learning is ubiquitous in ANN and DL domains, its use is still limited for SNNs. Stewart et al. [3] used transfer learning to teach the model new classes using few-shot learning.

3. METHODS

3.1. Introduction to DECOLLE

The DECOLLE ([2]) learning algorithm utilizes surrogate gradients with local learning rules, by using local readout matrices at every layer. The locality of the rules makes the algorithm hardware friendly and relatively more consistent with biology. The neuron and synapse model follows a leaky $I\&F$ equation with a relative refractory mechanism. The membrane potential U_i of a neuron i is defined as:

$$U_i(t) = V_i(t) - \rho R_i(t) + b_i, \tag{1}$$

$$\tau_{mem}\frac{d}{dt}V_i(t) = -V_i(t) + I_i(t), \tag{2}$$

$$\tau_{ref}\frac{d}{dt}R_i(t) = -R_i(t) + S_i(t). \tag{3}$$

where $S_i(t)$ is the binary value representing the presence or absence of a spike of neuron i at time t. A spike is emitted when the membrane potential exceeds a threshold. The constant b_i is the intrinsic excitability of the neuron. R_i captures the refractory dynamics. The neuron inhibits itself by weight ρ. τ_{ref} and τ_{mem} are time constants of the membrane and refractory dynamics, and I_i represents the synaptic current of neuron i.

In DECOLLE, a random readout is attached to each of the N layers of spiking neurons as:

$$Y_i^l = \Sigma_j G_{ij}^l S_j^l, \tag{4}$$

where G_{ij}^l are fixed random matrices, one for each layer. The global loss function is the sum of the layerwise loss functions on the random readouts, i.e. $\lambda = \Sigma_{l=1}^N L^l(Y^l)$. To enforce locality, all non-local gradients are set to 0.

The final DECOLLE learning rule is as follows:

$$\Delta W_{ij}^l = -\mu \text{error}_i^l \sigma'(U_i^l) P_j^l, \tag{5}$$

$$\text{error}_i^l = \Sigma_k G_{ki}^l (Y_k^l - \hat{(Y)}_k^l), \tag{6}$$

where $\hat{(Y)}^l$ is the pseudo-target vector for layer l.

For further information and derivations, refer to [2].

3.2. Quantization of Weights

We use quantization-aware-training, where during the forward pass, the weights are quantized before participating in computation. However, a full precision copy of the weights (i.e. shadow weights) are kept and the back-propagation is done through these full precision weights and the weight updates are applied. During inference, the network operates solely on the quantized weights and therefore, this method is really effective for offline training and online inference on hardware with limited resources. Thus, the weight update formula for QAT during training becomes:

$$w_q^{t+1} = Q(w_{fp}^t + \Delta w_q^t) \tag{7}$$

where w_q and w_{fp} are quantized and full precision shadow weights respectively. The weight quantization function Q is defined as follows:

$$Q(w) = \begin{cases} w_{min}, & \text{if } w \leq w_{min} \\ w_{max}, & \text{if } w \geq w_{max} \\ f(\frac{w}{S^\Delta}) \cdot S^\Delta, & \text{otherwise} \end{cases} \tag{8}$$

where w is the value of the weight, f is the rounding function and S^Δ is the step value. w_{min} and w_{max} are the min and max clamping levels, which are set separately for each layer as follows. The weight distributions during learning were plotted, to examine the values at which most weights lie. From these distributions, w_{min} and w_{max} were obtained, such that most weights before and after learning were between these values. Finally step-size S^Δ was determined as follows:

$$S^\Delta = \frac{w_{max} - w_{min}}{2^m} \tag{9}$$

where m is the bit precision of the weights. Moreover, while we vary the value of m to explore the accuracy-resource trade-off, the random readout weights G_{ij}^l are initialized as 2-bit weights throughout the experiments.

While we described the rounding function as f previously, the actual choice of the rounding function can have significant effect on the performance of the network. Therefore, we also test the weight quantization using a variety of rounding functions such as *ceiling*, *floor* and *round* functions. Furthermore, to examine the effect of the rounding function on the network convergence, we also analyze the weights of the network resulting from use of these rounding functions.

3.3. Transfer Learning

The QAT described in section 3.2 is less suitable for online training on neuromorphic hardware, because of the additional memory and computational overhead required for full precision shadow weights. Therefore, we propose a transfer learning framework where the full network is trained using QAT first and then, the last layer is retrained using only quantized weights (without any shadow weights), as illustrated in Fig. 1. Since training the entire network with quantized weights without any shadow weights would reduce the overall accuracy while implementing QAT on neuromorphic hardware increases overhead, the proposed method introduces a trade-off between the two.

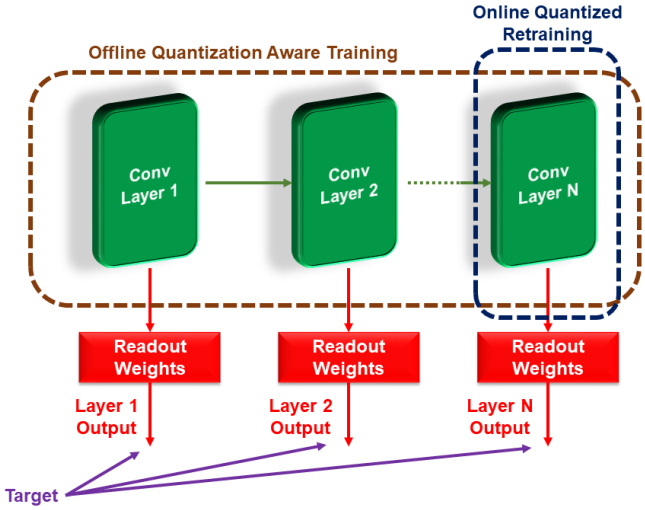

Fig. 1. Quantized Transfer Learning

However, one primary issue encountered during fully quantized training is that, if the weight update is smaller than the step-size, there will be no weight update. Therefore, if the upper limit of the weight update is ΔW_U and the step-size for a bit precision m is S^{Δ} and maximum and minimum weights w_{max} and w_{min} respectively, the necessary condition for the network to learn is:

$$\Delta W_U \geq S^{\Delta} = \frac{w_{max} - w_{min}}{2^m} \tag{10}$$

Therefore, the lower m is, the higher the ΔW_U is required for a network to learn. Hence, we use a learning rate modifier to modify the learning rate for different bit precision. This is a multiplicative learning rate modifier and is inversely proportional to the bit precision of the weights.

4. RESULTS

In this work, we use two datasets to experiment and evaluate the models, DVSGesture [14] and N-MNIST [15]. The DVS-Gesture dataset consists of 11 distinct hand gestures recorded using a DVS camera. For each gesture, there are 1342 samples recorded from 29 subjects. While for the original dataset the dimension of the frames were 128×128, we downsampled it to 32×32 and the temporal resolution used is 1 ms. The rest of the parameters were kept same as the original DECOLLE paper [2] for fair comparison. The N-MNIST dataset is the event based version of the more popular MNIST dataset. The dataset consists of 60000 training and 10000 testing images of handwritten digits $0-9$. This dataset was captured by recording MNIST images displayed on a screen using an ATIS sensor mounted on a pan-tilt unit ([15], [16], [17]) following predetermined sequences of saccadic motions. For this dataset too, preprocessing methods and parameters are kept same as Kaiser et al.[2].

For this work we use a 3 convolutional layer network architecture, similar to the one used in [2], but the major difference in ours is the absence of a fully connected layer at the end to reduce the number of parameters/computational complexity. Our experiments show that adding such a fully connected layer does not result in any significant improvement in classification performance.

4.1. Examination of rounding functions

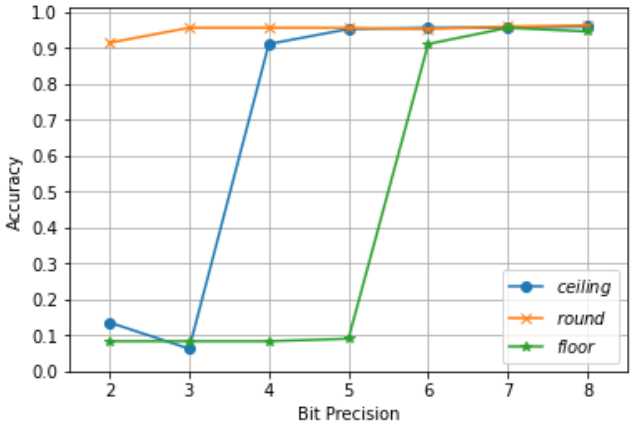

Fig. 2. Quantization with different rounding functions

In Section 3.2 we mentioned exploring effects of different rounding functions *ceiling*, *floor* and *round* on the performance of the network. In Figure 2, we see the results for these different quantization functions for DVS-Gesture dataset.

These different results are due to different distributions of trained weights - while $floor$ and $ceiling$ functions result in weights that are skewed to particular values, $round$ function results in weight values that are more evenly distributed. Weights therefore take on more values, resulting in better accuracy. More detailed discussions are out of scope of the current publication, and to be addressed in our future work.

4.2. Quantized Local Learning

In Table 1 we compare our results with other recent works namely, SLAYER [7], DECOLLE [2], Tandem Learning [18] and IBM Truenorth [14]. For DVS-Gesture dataset, the network is trained for 50 epochs. The quantized model's accuracy is slightly lower than full precision DECOLLE but higher than SLAYER while using only 3-bit weights. While Amir et al. [14] used ternary weights, our accuracy is higher than the TrueNorth based implementation. For N-MNIST dataset, our network reaches an accuracy of 98.75%, which is slightly lower than the previous works, but all others used 32-bit weights. Compared to original DECOLLE, our proposed model requires more than $10\times$ less memory and proportionally lower computational complexity but loses $< 0.3\%$ in terms of accuracy for both datasets.

Table 1. Comparison of Results

Dataset	Work	Weight Precision (# of bits)	Accuracy
DVS Gesture	Shreshtha et al. [7]	32	93.64%
	Kaiser et al.[2]	32	95.54%
	Amir et al.[14]	2	94.49%
	Our Work	**3**	**95.48%**
N-MNIST	Shreshtha et al.[7]	32	99.2%
	Kaiser et al. [2]	32	99.04%
	Wu et al.[18]	32	99.31%
	Our Work	**3**	**98.75%**

4.3. Quantized Transfer Learning

For quantized transfer learning, we used the same 3 layer convolutional network and trained all the layers using QAT with 2-bit precision. For DVS-Gesture dataset we used 6-5 way classification where the network is trained on 6 classes first using QAT and later the last layer is retrained with 5 other classes using fully quantized weights (i.e. without shadow weights). For both pre-training and retraining, the network is trained for 50 epochs. While the previous 2 layers weights remain at 2-bit precision, for the last layer retraining, the bit precision is varied from 2 to 6 bits.

The results are shown in Fig. 3. The network reaches 89.09% accuracy for 4-bit weights and 92.64% accuracy for 6-bit weights for the new 5 classes. For 6-bit, after epoch 1, 5 and 20, accuracy is 62.5%, 75.69% and 88.19% respectively. These results, despite with lower weight precision, are generally significantly better than the results obtained by [3] for 6-5 way classification using 8-bit weights, which are 64.7% with 1-shot, 65.1% with 5-shot, and 80.2% with 20-shot, where 1 shot is same as 1 epoch.

For the N-MNIST dataset, the network is trained for 5-5 way classification. For pre-training and retraining, the network is trained for 10 epochs. The network achieves 93.85% accuracy for 4 bits and 97.40% accuracy for 6 bits. While the accuracies do fall short of full precision accuracies, the results are promising considering the retraining is done completely on quantized weights.

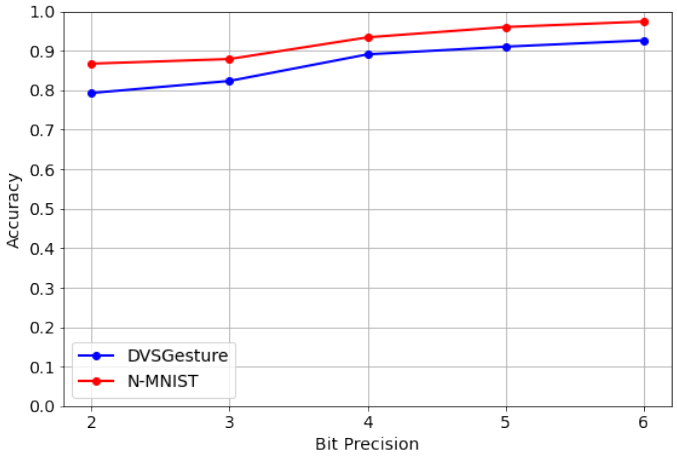

Fig. 3. Online Fully Quantized (no shadow weight) retraining on last layer For DVSGesture and N-MNIST

5. DISCUSSION AND CONCLUSION

Wu et al. [12] has explored quantization in progressive tandem learning for MNIST and CIFAR-10, where 4-bit on MNIST results in an accuracy drop of 0.03%, while our QAT 4-bit has an accuracy drop of 0.15%, both with respect to its own 32-bit baseline. Note however because N-MNIST is an event-based rendition of static MNIST images, and the number of timesteps per input sample is widely different: 16 after spike encoding for MNIST in [12], and 300 for N-MNIST, the results cannot be compared in a straightforward manner.

In summary, we have explored QAT to train SNNs with both bio-realistic and effective local learning rules, which to our knowledge is the first work of such kind. We achieve significant reduction in weight precision with minimal loss in accuracy for both DVSGesture and N-MNIST datasets. We propose a quantized transfer learning scheme that retrains online to learn new classes in fully quantized manner (no shadow weights) based on offline QAT trained network. Since a central goal of neuromorphic algorithms and systems is to achieve performance comparable to traditional systems at a fraction of memory/computational cost, we believe this work sets up a firm groundwork to design and deploy memory and power efficient hardware friendly SNNs.

While we have explored several different quantization functions and their effect on the performance, it is not yet conclusive. In the future, we plan to explore more advanced quantization schemes. Moreover, we also plan to test the proposed methods on more neuromorphic datasets such as ASL-DVS [19].

6. REFERENCES

[1] S. Yu, "Neuro-inspired computing with emerging non-volatile memory," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, Feb 2018.

[2] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," *Frontiers in Neuroscience*, vol. 14, no. 424, 2020.

[3] Kenneth Stewart, Garrick Orchard, Sumit Bam Shrestha, and Emre Neftci, "Online few-shot gesture learning on a neuromorphic processor," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 512–521, 2020.

[4] S. Thorpe, A. Delorme, and R.V. Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[5] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLOS Computational Biology*, vol. 3, no. 2, 2007.

[6] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, no. 99, 2015.

[7] Sumit Bam Shrestha and Garrick Orchard, "SLAYER: Spike layer error reassignment in time," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018),*, 2018.

[8] Yingyezhe Jin, Wenrui Zhang, and Peng Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7005–7015.

[9] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[10] Zilin Wang, Kefei Liu, Xiaoxin Cui, and Yuan Wang, "Deep spiking binary neural network for digital neuromorphic hardware," in *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*. IEEE, 2020, pp. 1–3.

[11] Gopalakrishnan Srinivasan and Kaushik Roy, "Restocnet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing," *Frontiers in neuroscience*, vol. 13, pp. 189, 2019.

[12] Jibin Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. Tan, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2021.

[13] Jyotibdha Acharya and Arindam Basu, "Deep neural network for respiratory sound classification in wearable devices enabled by patient specific model tuning," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 3, pp. 535–544, 2020.

[14] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al., "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[15] G. Orchard, G. Cohen, A. Jayawant, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, no. 437, 2015.

[16] L. R. Iyer and A. Basu, "Unsupervised learning of event-based image recordings using spike-timing-dependent plasticity," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1840–1846.

[17] L. R. Iyer and Y. Chua, "Classifying neuromorphic datasets with tempotron and spike timing dependent plasticity," in *2020 International Joint Conference on Neural Networks (IJCNN) (accepted)*, 2020.

[18] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[19] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos, "Graph-based spatio-temporal feature learning for neuromorphic vision sensing," *IEEE Transactions on Image Processing*, vol. 29, pp. 9084–9098, 2020.