Dynamic Precision Tunability in Low-Power Training of Deep Neural Networks

Shayan Moini, Matthew Caswell, Wayne Burleson
Department of Electrical and Computer Engineering, University of Massachusetts Amherst
Amherst, Massachusetts, 01003, USA
(smoini, mcaswell, burleson)@umass.edu

ABSTRACT

Deep neural networks (DNN) have been utilized in numerous machine learning problems due to their high inference and generalization capabilities. Their signature features including their high computational complexity and their hunger for large datasets have made Graphical Processing Units (GPU) a suitable platform for accelerating both their training and inference. However, semi-custom IP blocks for ASICs and FPGAs now compete with GPUs in terms of higher performance per watt and their ability to perform computations with variable number formats and word lengths. Our work focuses on studying the effects of different number representation on the training performance of DNNs. More specifically, we are trying to replace the floating point number representation with less accurate representations that tend to decrease the computational complexity, memory/bandwidth requirements, and power consumption of DNN training while approximating the original full precision classification accuracy of the final trained model. The goal is to dynamically tune the number of bits in fixed-point number representation during the training process and generate a hardware accelerator that can adapt to this dynamic change and utilize it for reducing power consumption. As an initial study of the impact of the dynamic fixed-point number representation, we have conducted a two-phase training of a 6-layer DNN on the CIFAR-10 dataset with fixed-point calculations to measure classification performance, and an ASIC/FPGA-based model for power consumption.

KEYWORDS

Deep Learning, Custom Number Representation, Training, Neural Network

1 PREVIOUS WORK

Recently, great effort has been made to replace the traditional single-precision floating point number representation with lower precision alternatives. The main benefits of low precision computation are lower power consumption, lower memory and bandwidth requirements to load, store, and transfer the weights and activations from off-chip memory, and higher clock speed because of the lower computational complexity of low precision number representation compared to single precision floating point. This effort has been challenging compared to quantizing DNN inference due to the high dynamic range of weight gradients generated during the back-propagation training algorithm. In one of the first efforts to eliminate floating point representation, Authors of [2] have shown that if a DNN is trained with fixed-point number representation and using stochastic rounding, the classification performance of the final model is very similar to the full-precision model. However, this work has only focused on shallower DNNs and cannot be extended to deeper ones (for example 7 or 8 layer networks).

Another method, High Accuracy Low Precision Training (HALP) for quantizing the computations in Deep Neural Networks is discussed in [1]. A hybrid method is used; at each epoch, an initial floating point back-propagation operation is performed and the rest of the operations of the epoch are performed using fixed-point representation. To deal with the large dynamic range of gradients during training, two methods are proposed. Stochastic variance reduced gradient (SVRG) is introduced for reducing the gradients, dynamic range, and increasing convergence speed. Bit re-centering is used at each stage to determine a scaling factor for storing all the gradient values in fixed-point representation. We conducted a number of different experiments using this algorithm [7] with different fraction widths performed on a ResNet-18 network trained on the CIFAR-10 dataset. As shown in Fig. 1, the trained network maintains its performance until 4 bits of fraction width. However, this approach in its current form is impractical for large datasets since it needs orders of magnitude higher memory storage space to store all the intermediate gradients for normalization at the end of each epoch.

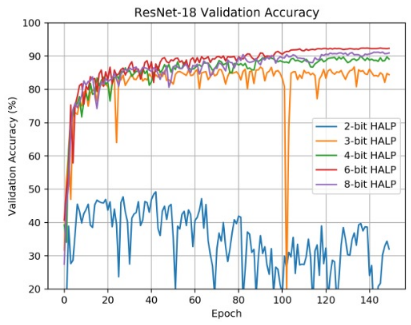

Figure 1: HALP training results with different bit-widths

A new number representation called Flexpoint for training DNNs is introduced in [4]. This number representation consists of a 16-bit fixed-point number representation as an independent mantissa and 5 extra bits as a shared exponent between all the weights in a layer. Compared to a full precision floating point implementation, this representation halves the required memory to store the weights of DNN, and by making sure all the operations are performed in fixed-point, it decreases the computational complexity and subsequent power consumption of the DNN operations. The Flexpoint scheme with 16 bits of mantissa and a 5 bit shared exponent can match the training accuracy of full precision floating point on multiple different state-of-the-art deep networks.

Dynamic precision scaling has previously been addressed in [6]. However, in [6], the precision steps are large (training is toggling between 16/8 bit or 32/16 bit fixed-point representations) and the DNNs that have been studied are very small MNIST DNNs. They have also not reported the power benefits of dynamic precision tuning compared to a steady high-precision scheme.

2 DYNAMIC PRECISION TUNING AND BLOCK FLOATING POINT

In the present work, all the operations of DNN training are performed using fixed precision number representation. The main idea is that the dynamic range of gradients is not the same during the whole process of training, at the beginning of the training, gradients are large and the weights change dramatically after each epoch while at the end of the training process, the gradients get very small and the weight changes are very subtle. Using this insight, we can monitor the training process and tune the precision of the model at each stage to the lowest precision possible.

The main benefit of this method is the lower communication bandwidth between the processing unit and the off-chip memory during low-precision training epochs which in turn helps with the overall clock speed of the DNN training considering it is a bandwidth-intensive operation. Using smart design practices, the power consumption of the DNN accelerator can also be dynamically tuned based on the chosen bit-width at each stage. The shorter bit-widths result in lower switching activity and power consumption. 2 shows the result of an experiment on a ResNet network trained on the CIFAR-10 dataset. The low precision DNN training was simulated using a custom fixed-point library developed for Keras [5] and it was run on GPUs provided to UMass Amherst Faculty and Students by the Massachusetts Green High-Performance Computing Center (MGHPCC). The bar figure shows the estimated application specific hardware power consumption calculated as rough measures based on [3]. The lines show training validation accuracy for different number representations. The dynamic scheme uses 8-bit fixed precision representation for the first half of training and full-precision for the second half of training. As shown in 2, the hybrid scheme has better power consumption compared to full-precision with a slight degradation in accuracy.

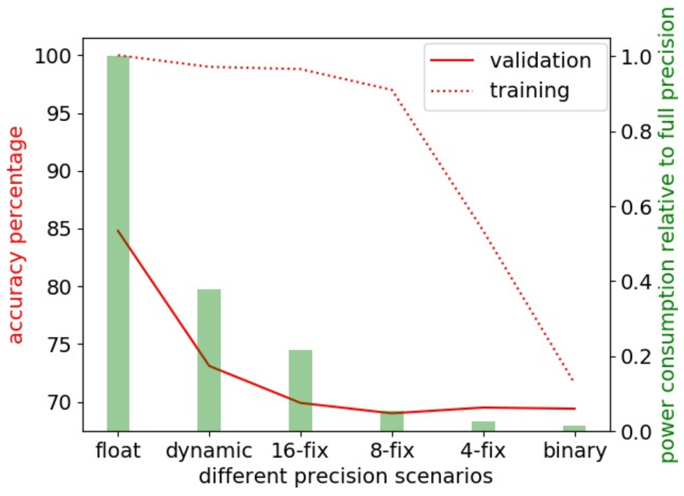

Figure 2: HALP training results with different bit-widths

3 LOOKING FORWARD

Based on our experimental method, we can explore a number of extensions to this work. One of our goals is to find metrics during the training process that help in determining the best epoch to scale up or down the training precision. The current metric of training accuracy is not necessarily the most informative metric compared to alternatives such as statistical metrics of weight gradients including minimum absolute value, the standard deviation of the first quantile, and maximum value divided by minimum value.

Additionally, when migrating to deeper networks for larger datasets, we may face situations where the dynamic range of even the largest fixed-point number is not enough to cover all the gradients. Here we can switch to a shared exponent floating point number representation similar to Flexpoint and extend the idea of dynamic precision tuning to the mantissa of FlexPoint representation.

We are also working on developing a custom hardware architecture supporting different widths for fixed-point training of DNN which can show better power and performance when switched to lower bit-widths. The run-time power measurements from this architecture can replace power estimations made in the previous section.

REFERENCES

[1] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. 2018. High-Accuracy Low-Precision Training. *arXiv preprint arXiv:1803.03383* (2018).
[2] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*. 1737–1746.
[3] Mark Horowitz. 2014. 1.1 computing's energy problem (and what we can do about it). In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*. IEEE, 10–14.
[4] Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K Bansal, William Constable, Oguz Elibol, Scott Gray, Stewart Hall, Luke Hornof, et al. 2017. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Advances in Neural Information Processing Systems*. 1742–1752.
[5] Bert Moons. 2018. Quantized Neural Networks - networks trained for inference at arbitrary low precision. (2018). https://github.com/BertMoons/QuantizedNeuralNetworks-Keras-Tensorflow
[6] Taesik Na and Saibal Mukhopadhyay. 2016. Speeding up convolutional neural network training with dynamic precision scaling and flexible multiplier-accumulator. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 58–63.
[7] Chris Re. 2018. HALP : High Accuracy Low Precision Training. (2018). https://github.com/HazyResearch/torchhalp