#101

[R1]

This paper describes an accelerator for CNN training that incorporates early exploration of sparsity and data quantization.

This goal is a good one. Inference is clearly an FPGA strong point, but there are limited papers on training. And combining quantization and sparsity would seem like a good match for an FPGA.

I think the actual contributions of this paper are to explain the approaches to sparsity and accumulation during back propagation. The results are not superior to [4], which the authors admit.

I found the treatment of bit width very confusing. I do not understand how this is handled dynamically. The slides in Figure 11 imply that it is not dynamic: you train either for 32-bit, 24-bit or 16-bit. I also found that Figure 11's legend was meaningless. I don't know what "p120" means, for example.

I also found Figure 6 important but illegible (without magnifying the PDF) and when I could, I didn't really understand the legend.

While the paper is very well organized and generally well-written, the grammar needs work. I think a professional English-language proofreader should be used if this paper is selected for publication.

[R2]

This paper proposed an FPGA solution for training CNNs. This solution exploits sparsity and low precision features to improve performance. Dedicated PEs are designed to support both operation-sparse and result-sparse patterns. Besides, dynamic loop mapping strategy is employed to further improve performance. Experiments using VGG-11 demonstrate high performance and energy efficiency of this proposed solution.

CNN inference with FPGA-based accelerators has been well studied in recent years, while only a few works focus on CNN training. This work tries to leverage sparsity and low precision features to improve training efficiency. Especially, when pruning, both operation-sparse and result-sparse patterns are exploited for forward and backward training phases.

However, the experiment evaluation is very poor. Only one CNN (VGG-11) and one dataset (CIFAR-10) are evaluated, which is not convincing. Also, VGG-11 is out-of-date. More state-of-the-art benchmarks and datasets should be used for evaluation, such as ImageNet, ResNet, googlenet, etc.

The technique details are not clearly discussed. For different training stages and phases, distinct precision, loop mapping, and sparsity are used. For example, at first, 32-bit single precision data are used to train a full precision model, while 8-bit fixed point data are used later when fine-tuning. Besides, different phases use distinct loop mappings. How does the proposed architecture handle this? Or if reconfiguration is required, what is the overhead?

Currently, this work requires to train a full precision CNN as its first step. It would better if the early pruning challenge could be further addressed since training a fully precision CNN on FPGAs is very expensive.

There exist some format errors and typos, such as the missing section 4.1 and missing blank.

[R3]

This paper presents a low-precision, sparse neural network training accelerator on FPGA. The overall training process is as follows: (1) train the model in floating-point on GPU for some epochs; (2) prune and quantize the network and continue training the sparse network on FPGA. The pruning and quantization techniques are leveraged from existing works. Pruning is done for entire k-by-k conv filters --- this means the DNN is dense in the spatial dimensions but sparse in channel dimensions. A sparse matrix representation and compute engine is still necessary. The HW accelerator processes a DNN one layers at a time, storing output activations off-chip for reuse during backprop update. Because of sparsity in the channels, the authors focused on exploiting parallelism in batch, spatial, and kernel dimensions. The hardware architecture makes use of techniques such as sparse matrix-vector mult, memory banking, and loop unrolling. The HW achieves 90% utilization on most conv layers of a pruned VGG-11 network. The main contributions of this paper is that it brings together low-precision, sparsity, training acceleration, and makes it work on a real FPGA platform. Each individual technique is not novel, but this is the first FPGA accelerator for sparse low-precision training of a neural network.

Strengths:

* First example of sparse low-precision training accelerator on FPGA. In general there are very few such works in hardware architecture conferences.
* The training and pruning process is practical and incorporates FPGAs in a smart way.
* Strong comparison versus SOTA DNN accelerators on FPGA.

Weaknesses:

* The training is only done on just one network. Since this is a training instead of an inference accelerator, the authors should provide results on various network to demonstrate the effectiveness and generalization of this technique.
* Evaluation done with an ImageNet model (VGG-11) trained on CIFAR-10. This means the model is highly overparameterized likely making it easy to prune while maintaining high accuracy. Also, CIFAR-10 images are small meaning that storing the activations (a big challenge for DNN training) is much easier compared to training on ImageNet. The results in Table 5 should be worse on a more practical CIFAR-10 model (less spatial parallelism to extract) or a more practical ImageNet model (larger activation size).
* A more practical network will present challenges which reduce the hardware performance and energy efficiency.
* In Table 5, the paper didn’t list the power efficiency of the whole FPGA+GPU system. In addition, the power of GPU listed is the maximum TDP power from the datasheet. The authors didn’t measure the practical power of training on GPU. Training on CIFAR-10 dataset may not cost that much power.

[R4]

This paper proposes an accelerator for accelerating the fine-tuning training process of sparse (pruned) neural networks.

In my view, the most important contribution of the paper is the proposed design for performing both forward and backward pass for sparse weights using the same datapath. This poses a design problem and the proposed solution (facilitated by using CSB to make both non-transpose and transpose accesses) is (to my knowledge) a novel and interesting one. However, considering the paper from a CNN training perspective, there are major gaps that make the content questionable:

1)    The proposed training accelerator works only on already-pruned (sparse) neural networks, which implies that the network was trained as a dense network in the first place. This implies the accelerator is not accelerating the part of training that actually needs acceleration. Another option would have been to consider a priori (fixed prior to training start) sparsity, but this is not mentioned as an option in the paper.

2)    The effects of precision on training accuracy is not discussed or explored. Training on datasets like ImageNet with 8-bit arithmetic is an open problem and often requires advanced techniques to converge, see e.g. http://export.arxiv.org/pdf/1805.11046

3)    How the backward pass is implemented for error calculation, activations, pooling layers, batch normalization etc. is not discussed

4)    How the solution handles framework integration, data preprocessing etc. is not discussed.

5)    Results from only a single network on a single dataset is presented.

Other problems I observed with the content:

•    The transposed full convolution approach for convolution backward pass (Section 2.2) works for stride=1 but is not straightforward for other cases and not always applicable. Larger strides require specialized padding or col2im. How do you handle stride>1?

•    Figure 5 is the core of the paper needs a more details and explanation. How are weight updates handled?

•    Section 4.1 is empty (the text seems to be merged into 4.2?)

•    The precision is explained to be “block fixed point” in 4.2 where the shared exponent is a power of two. How many bits are used for the exponent? How does this impact accuracy?

•    The results in Section 6.1 and Figure 11 are not really relevant to the content of the paper, as this part of the training is not accelerated by the proposed accelerator.

My recommendation would be to focus on the core contribution and present this as a poster.