ACCELERATING DEEP CONVOLUTIONAL NETWORKS USING LOW-PRECISION AND SPARSITY

Ganesh Venkatesh, Eriko Nurvitadhi, Debbie Marr

Accelerator Architecture Lab, Intel Corporation

ABSTRACT

We explore techniques to significantly improve the compute efficiency and performance of Deep Convolution Networks without impacting their accuracy. To improve the compute efficiency, we focus on achieving high accuracy with extremely low-precision (2-bit) weight networks, and to accelerate the execution time, we aggressively skip operations on zero-values. We achieve the highest reported accuracy of 76.6% Top-1/93% Top-5 on the Imagenet object classification challenge with low-precision network while reducing the compute requirement by $\sim$3$\times$ compared to a full-precision network that achieves similar accuracy. Furthermore, to fully exploit the benefits of our low-precision networks, we build a deep learning accelerator core, DLAC, that can achieve up to 1 TFLOP/mm^2 equivalent for single-precision floating-point operations ($\sim$2 TFLOP/mm^2 for half-precision), which is $\sim$5$\times$ better than Linear Algebra Core [16] and $\sim$4$\times$ better than previous deep learning accelerator proposal [8].

***Index Terms*—** Deep Neural Networks, Ternary-weight Convolutions, Accelerator

1. INTRODUCTION

Deep Convolutional Neural Networks (DNNs) provide state-of-the-art accuracy for many computer vision and image analysis tasks [13]. The accuracy of DNNs is rapidly improving (for example, Top-5 error for Imagenet object classification challenge [9] improved by >6$\times$ in 5 years) and is close to human-level accuracy in some cases [12].

DNNs achieve higher accuracy by building more powerful models consisting of greater number of layers (network depth). However, this increase in network depth incurs a steep increase in compute and memory requirements. As a result, these networks are taking longer to train; multiple week training times are common even when using multiple GPU cards. Also, the greater compute requirements make DNNs harder to deploy, which has led to a lot of interest recently in specialized hardware solutions, both commercially [3, 5] and in academia [10, 6, 8].

In this paper, we build on recently proposed low-precision convolution networks [17, 15, 14] to reduce compute requirements of these networks. While previous efforts compromise accuracy to gain compute efficiency, we aim to achieve similar (or slightly better) accuracy at a lower compute complexity. In particular, we train a low-precision variant of a 34-layer deep residual network (Resnet [13]) that attains higher accuracy than the *vanilla* 18-layer deep resnet while requiring fewer floating-point operations ($\sim$3$\times$ lower) and having a smaller model size (7$\times$ smaller). Furthermore, to fully leverage the benefits of this low-precision network, we propose and evaluate a Deep Learning Accelerator Core, DLAC, that can achieve equivalent performance of up to $\sim$1 Teraflop/mm^2 by skipping operations on zero values. We make the following contributions in this paper

Demonstrate high accuracy using low-precision weights Using low-precision 2-bit weight networks [14], we achieve high accuracy of 76.6% Top-1/93% Top-5 on Imagenet [9], ***the highest reported with a low-precision network to our knowledge*** and within 1.3% of the 2015 Imagenet winner [13]. Furthermore, we show that in these low-precision networks, most of the floating-point operations operate on zero values – both while training as well as inference.

Define a deep-learning accelerator core,DLAC We propose a deep-learning accelerator core, DLAC, that exploits the available sparsity in these networks to achieve high effective performance and can be applied to both training as well as inference.

Achieve high effective performance density of $\sim$1 TFlop/mm^2 Our evaluation, based on synthesis of our design in 14nm, shows that DLAC can sustain extremely high performance density, reaching up to 1 TFlop/mm^2 equivalent performance for many of the layers in current state-of-the-art Residual networks [13]. This is an order-of-magnitude higher performance density than the previously proposed deep learning accelerator [8].

2. LOW-PRECISION DEEP CONVOLUTION NETWORK

We first overview recent efforts on using low-precision weights as well as inducing sparsity in deep convolution networks, then motivate the need for our work. We then present how we train/fine-tune low-precision networks. We conclude this section with analysis on the sparsity available in these networks, quantifying the potential of acceleration

by a hardware architecture that is optimized for efficient zero-skipping.

2.1. Background: Lowering the precision of network weights

Several techniques have been proposed to lower the precision of network weights. These include approaches that reduce the weights to half-precision [7], binary value [17], and ternary (2-bit) value [14]. While reducing the precision from float to half-precision can get almost 2× savings, the other approaches can achieve significantly higher savings (16-32× smaller model, very few float multiplications). We use the approach from Lin et al. [14] that lowers the network weights to a ternary value with the options being `[-1, 0, 1]` as follows:

$$W_{ter}(i,j,k,l) = \begin{cases} 1 & : \text{if W(i,j,k,l)} > w_{th} \\ -1 & : \text{if W(i,j,k,l)} < -1 * w_{th} \\ 0 & : \text{otherwise} \end{cases} \quad (1)$$

While the previous proposals trade classification accuracy to achieve compute efficiency, we focus on low-precision networks that provide similar (or slightly better) accuracy than a full-precision network while requiring lesser compute.

The recent DeepCompression [11] proposal looks at inducing sparsity in deep convolution networks via *pruning* where certain weights/activations are clamped to zero value. Instead, we explore techniques that induce *dynamic* sparsity in networks where the zero weights/activations can change across input samples and training phases. By doing so, the effective sparsity we achieve is much higher (>2×) and we show its applicability on state-of-the-art networks.

2.2. Evaluation

We evaluate on two datasets - Cifar10 and Imagenet

Cifar10 [2] This dataset consists of 60K small images from 10 different classes and we train on them using VGG [18] (baseline training recipe from [1]).

Imagenet [9] This dataset consists of 1.2 million images for training and 50,000 images for testing from 1000 classes. We train on this dataset using the Residual networks, Resnet [13] (baseline training recipe from [4]).

We implement our proposals in Torch, a deep learning framework and our accuracy evaluations use single-crop testing. For evaluating our deep learning accelerator, we synthesize it in 14nm Intel process.

2.3. Training Low-precision Networks

We experiment with a number of different approaches to improve the accuracy and reduce the compute intensity of these extremely low-precision networks. In this section, we overview the different techniques -

Pre-initialization from full-precision network We train the network in full-precision for the first few iterations (15 iterations) and then switch over to the low-precision mode for rest of the training (∼75 iterations). This improves the accuracy by almost 2% (Section 4.2).

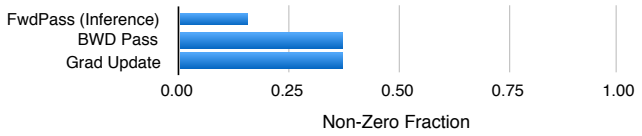

Fig. 1. **Sparsity in Low-precision Networks** Graph plots the fraction of operations on non-zero operands when training Resnet-34 (34-layer deep) on Imagenet.

Skipping lowering of precision for parts of network We do not lower the precision of the first layer in the network to minimize the information loss from the input image. This improves our Top-1 accuracy by ∼0.5% (Section 4.1).

Aggressive lowering of learning rate We maintain a history of the train error and lower the learning rate when the train error does not go down for few iterations. This can improve our Top-1 accuracy by more than 1% in some instances.

Regularization We regularize activations to reduce noise and induce more sparsity. As a side effect, this technique tends to smooth our convergence curve (Figure 3).

ReLU threshold We vary the thresholds on our rectifier units to induce higher sparsity and reduce noise.

2.4. Sparsity in Low-precision Networks

In this section, we show that the vast majority of the operations in low-precision networks operate on zero values and hence, we can further reduce the compute requirements of these networks by skipping over operations on zero values (zero-skipping). The source of zero values in our network are as follows (i) **Rectifier Units (ReLU):** The rectifier units zero out activations below the threshold value (0.01), (ii) **Lowering the precision of weights:** The formula for lowering the precision effectively zeros out weights with small values (Eq. 1).

The amount of sparsity is shown in Figure 1 for low-precision Resnet-34 training on the Imagenet dataset. The data shows that a small fraction (16%) of operations during the forward pass (inference) operate on non-zero operands and only around 33% of operations during the backward pass operate on non-zero operands. This shows that we can improve performance by 3-6× by skipping operations on zero values.

3. DEEP LEARNING ACCELERATOR

DLAC is a two-dimensional grid of processing elements with buffers for network weights and input feature map (Figure 2). The processing elements have arithmetic units, buffers for output feature map and control logic for skipping over operations on zero values to leverage the sparsity in low-precision networks. To enable skipping of zero operations, we assign

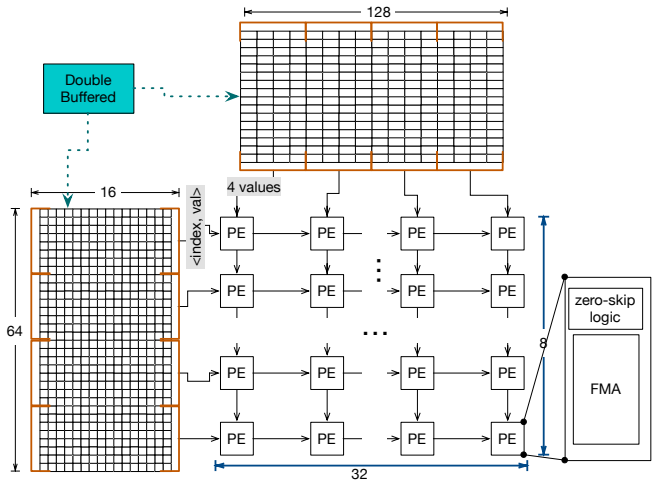

Fig. 2. DLAC **Architecture** It is 2-D grid of processing elements where each one has floating-point units and logic to perform zero-skipping

multiple output buffers to each processing element (trading for greater number of buffers over few more arithmetic units) and when scheduling operations to update these output buffers, we skip the ones operating on zero values to accelerate the performance.

Our accelerator supports the common operations in deep networks as follows:

`mmOp`: Matrix Multiply We use this to perform convolutions and the fully-connected layers.

`ptWiseOp`: Pointwise Operation The accelerators supports performing pointwise operations of the form <op, val1, val2, val3> on some or all the elements of the output buffer. The *op* can be arithmetic operations (add/sub, mul-add), or a ternary control expression (?:). We use this to execute the non-linearity (Relu) and batch normalization (inference).

DLAC **for training networks:** In training mode, we instantiate each DLAC with 512 single-precision floating-point units and all our datapaths are 32-bit wide. This allows DLAC to support the current standard training technique of using single-precision operations. Furthermore, our accelerator is able to sustain higher effective performance (3-4 TeraOps/cycle) by leveraging the *dynamic* sparsity in these networks.

DLAC **for inference:** In the inference mode, we instantiate each DLAC with 256 half-precision floating-point units, 256 half-precision adders and all the datapaths are 16-bit wide. This provides our accelerator with $\sim$2$\times$ performance density boost. Similar to the training phase, the DLAC is able to sustain much higher throughput because of efficient zero-skipping. In addition to the low-precision networks, DLAC can also accelerate pruned networks [11].

4. RESULTS

In this section, we evaluate the accuracy of our extremely low-precision networks as well as the performance they can attain on our deep learning accelerator, DLAC.

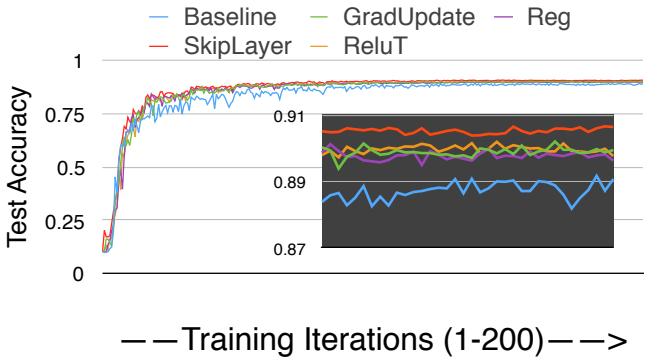

Fig. 3. **VGG Convergence Graph for Cifar10** The dark graph inside is zoomed-in version of the final-few epochs of training.

4.1. Accuracy of Low-precision Networks on Cifar10

We first evaluate low-precision variant of VGG network on Cifar10 dataset, shown in Figure 3. Our baseline network replaces the regular convolution operation with its low-precision variant, which reduces the accuracy by $\sim$3%. To improve the accuracy, we utilize the following techniques – (i) *Reg series*: we apply l1 regularization of activations which improves the accuracy by a small amount, (ii) *GradUpdate series*: to reduce overfitting, we backpropagate with zero error for samples that are classified correctly by the network and skip backpropagation step completely if the complete batch was correctly classified, and (iii) *ReluT series*: we change the rectifier unit threshold (ReluT series) to 0.01 for the last few epochs of training to reduce the noise in activations, and (iv) *SkipLayer series*: we compute the first convolution layer of the network in full-precision and convert the rest to low-precision variant. By doing so, we ensure that our model is able to capture all the information from the input image in the first layer while still benefiting from the efficiency of lower precision in the other layers of the network. The results show that these techniques improve the accuracy by $\sim$1.6%.

4.2. Accuracy of Low-precision Networks on Imagenet

In this section, we present the accuracy results on Imagenet dataset using multiple Resnet networks (2015 winner of Imagenet competition). We employ techniques that showed promise on Cifar10 dataset (regularization, ReLU threshold, skip precision lowering of first layer) and an additional trick of aggressively lowering the learning rate when the training accuracy stops improving.

Lowering precision of trained networks We start with the trained model of different Resnet networks, lower the precision of all the layers except for the first one, and train the resultant network. We report the accuracy numbers for the full-precision network and our low-precision 2-bit variant in Table 1. The data shows that low-precision networks provide better accuracy as the network depth increases. Hence,

Network Depth	Full-precision		2-bit precision	
	Top-1 [4]	Top-5 [4]	Top-1	Top-5
Resnet-18	69.56	89.24	-	-
Resnet-34	73.27	91.26	71.6	90.37
Resnet-50	76	93	73.85	91.8
Resnet-152	77.84	93.84	76.64	93.2

Table 1. Accuracy of Resnet network on Imagenet dataset for different depths (first column suffix), regular full-precision, and the extremely low-precision 2-bit version.

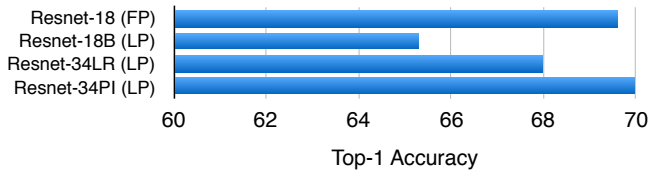

Fig. 4. **Training Resnet-34 with low-precision weights** The graph shows the accuracy we obtain by training Resnet-34 using low-precision 2-bit weights by aggressively lowering the learning rate (*Resnet-34LR*) as well as pre-initializing with a full-precision network(*Resnet-34PI*). We obtain higher accuracy than previous work on a low-precision network [14] (Resnet-18B) as well as the full-precision Resnet-18 while requiring fewer computations than either.

just as their full-precision counterparts, the low-precision networks can also scale their depth to provide higher accuracy. Based on the data in Table 1, we also observe that the low-precision variant of a larger network provide better accuracy than the regular full-precision network (low-precision Resnet-34/Resnet-50/Resnet-152 has higher accuracy than full-precision Resnet-18/Resnet-34/Resnet-50 respectively). This is an important result because the low-precision variant of a larger network needs less compute and has a smaller model size than the regular full-precision network. Hence, in effect, using the lower precision variant of a larger network achieves better accuracy and requires less compute than the original full-precision network.

Training low-precision Resnet We train Resnet with 34-layers and 2-bit weights. We try the following two techniques to improve accuracy (i) lowering the learning rate aggressively and (ii) training in full-precision for the first few iterations and switching over to low-precision after that. Our results are shown in Figure 4 – we attain ∼4.8% higher accuracy than previous work [14] using low-precision as well as slightly better accuracy than the regular full-precision variant of 18-layer Resnet.

4.3. Performance of DLAC on Low-precision Networks

Figure 5 shows the performance our accelerator can sustain for different convolution layers in Resnet-34. The accelerator can sustain up to 5K FLOP/cycle (2.78K FLOP/cycle on average), which at 500MHz translates to 2.5 Teraflops/second (1.34 Teraflops/second). Furthermore, the graph shows that

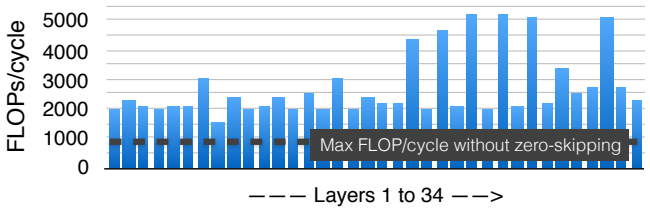

Fig. 5. **Performance of DLAC on Resnet-34** The data shows the performance our accelerator can sustain for each layer in 34-layer deep Resnet. The graph shows that our accelerator gets significant performance boost (1.8 - 5×) by skipping operations on zero-values and that our accelerator provides greater speed-up as we go deeper in the network because the layers get more and more sparse.

our accelerator provides better performance for the deeper layers of the network because of the greater sparsity in these layers. As a result, as we map deeper networks to our accelerator, we expect to get better performance because of its ability to exploit sparsity in the deeper layers. In single-precision mode, DLAC synthesizes to 2.2 mm^2 (1.09 mm^2 in 16-bit mode) cell area in 14nm with pure ASIC flow for all the buffers and the arithmetic units (no optimized macro-blocks). So the DLAC compute IP has compute denstiy of 0.6 Teraflop/s/mm^2 and can exceed 1 Teraflop/s/mm^2 for deeper layers in the network.

Comparison to prior work In comparison to DaDianNao Supercomputer [8], one DLAC instance provides similar to slightly better performance than one node of DaDianNao while being >4× smaller. Furthermore, our accelerator can provide higher performance as the sparsity in the network increases. In comparison to a recent work on zero-skipping [6], we achieve higher speed-ups by exploiting sparsity in both activations and weights. Unlike, EiE [10], we primarily focus on the convolution layers and support single-precision for the training phase.

5. CONCLUSION

In this paper, we looked at improving the accuracy of extremely low-precision DNNs and encouraging greater dynamic sparsity in these low-precision networks to reduce their compute requirements. To fully leverage the efficiency of these low-precision networks, we developed and evaluated a deep learning accelerator, DLAC, that sustains high effective flops by skipping over operations on zero values. We demonstrate that these low-precision networks can attain high accuracy, achieving 76.6% Top-1/93% Top-5 accuracy on Imagenet and that the low-precision variant of a larger network can possibly achieve higher performance while requiring lesser compute than a regular full-precision network. Our DLAC evaluation shows that we can sustain up to 1 Teraflop/mm^2 equivalent which is a significant improvement over previous accelerator proposals.

6. REFERENCES

[1] 92.45% on cifar-10 in torch. http://torch.ch/blog/2015/07/30/cifar.html.

[2] The cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.

[3] Tensor processing unit. https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html.

[4] Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html.

[5] Wave dataflow engine. http://wavecomp.com/technology/.

[6] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing.

[7] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *ArXiv e-prints*, December 2015.

[8] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, pages 609–622, Washington, DC, USA, 2014. IEEE Computer Society.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[10] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network. *ArXiv e-prints*, February 2016.

[11] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *ArXiv e-prints*, October 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *ArXiv e-prints*, February 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. *ArXiv e-prints*, March 2016.

[14] F. Li and B. Liu. Ternary Weight Networks. *ArXiv e-prints*, May 2016.

[15] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio. Neural Networks with Few Multiplications. *ArXiv e-prints*, October 2015.

[16] Ardavan Pedram, Robert A. van de Geijn, and Andreas Gerstlauer. Codesign tradeoffs for high-performance, low-power linear algebra architectures. *IEEE Trans. Comput.*, 61(12):1724–1736, December 2012.

[17] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*, 2016.

[18] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, September 2014.