FIXED POINT OPTIMIZATION OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT RECOGNITION

Sajid Anwar, Kyuyeon Hwang and Wonyong Sung

Department of Electrical and Computer Engineering
Seoul National University
Seoul 151-744 South Korea
Email: sajid@dsp.snu.ac.kr, khwang@dsp.snu.ac.kr, wysung@snu.ac.kr

ABSTRACT

Deep convolutional neural networks have shown promising results in image and speech recognition applications. The learning capability of the network improves with increasing depth and size of each layer. However this capability comes at the cost of increased computational complexity. Thus reduction in hardware complexity and faster classification are highly desired. This work proposes an optimization method for fixed point deep convolutional neural networks. The parameters of a pre-trained high precision network are first directly quantized using L2 error minimization. We quantize each layer one by one, while other layers keep computation with high precision, to know the layer-wise sensitivity on word-length reduction. Then the network is retrained with quantized weights. Two examples on object recognition, MNIST and CIFAR-10, are presented. Our results indicate that quantization induces sparsity in the network which reduces the effective number of network parameters and improves generalization. This work reduces the required memory storage by a factor of 1/10 and achieves better classification results than the high precision networks.

Index Terms— convolutional neural network, quantization, word length optimization, sparsity

1. INTRODUCTION

Convolutional neural network (CNN) is a neuromorphic computational model inspired from mammal's brain. The main inspiration comes from Hubel and Wiesel's work. Experiments on monkeys revealed that the visual area contains two types of cells: simple cells responsible for feature extraction and complex cells combining these features locally [1]. Yann Lecun et al. developed handwritten digit recognition CNN called LeNet-5 [2]. A sample CNN network is depicted in Fig. 1. CNN has three types of layers: convolution, pooling and fully connected DNN layers. Convolution layers exhibit local connectivity and weights sharing. The pooling layer performs subsampling on k x k (e.g., k = 2) region from the preceding layer. The operation can be average, max or stochastic pooling [2] [3] [4]. Conceptually CNN can be divided into two parts. The frontal part learns useful features for classification while the rear part consists of a multilayer fully connected deep neural network (DNN).

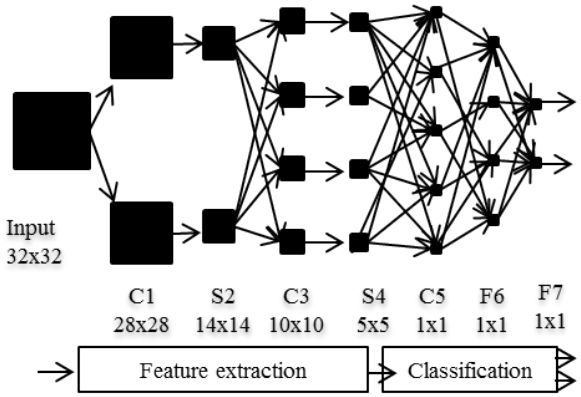

Fig. 1. Convolutional neural network with eight layers. The prefix "C", "S" and "F" stands for convolution, subsampling and classifier layers respectively. Layers C1 to C5 and C5 to F7 constitute the frontal feature extraction and classifier parts respectively. The proposed work represents the depicted architecture with a string of 1-2-2-4-4-5-4-2 where each number denotes the count of feature maps in that layer.

CNN is widely used for handwritten digit recognition [2] and general object recognition [5] [6] [7]. Alex et al. used CNN to classify 10, 100 and 1000 objects [5] [6]. They used a big network with 8K convolution connections. Network parameters have 32-bit floating point (single) precision. Therefore reducing the word length is highly desired, especially for VLSI implementation.

In the literature, fixed point implementations of neural networks have been studied. A quantized feed forward deep neural network is introduced in [8] and [9]. However CNN

This work was supported in part by the Brain Korea 21 Plus Project and the National Research Foundation of Korea (NRF) grants funded by the Ministry of Education, Science and Technology (MEST), Republic of Korea (No. 2012R1A2A2A06047297).

has more diverse layer types and hence quantization is more challenging. The work of [10] uses a directly quantized CNN. However it does not provide a retraining mechanism with low precision weights.

The proposed work has two important contributions. We provide a training mechanism with quantized weights which reduces the cost of VLSI hardware implementation. Weights with floating point precision are reduced to 3 and 4-bit precision, which yields more than 80% savings. Secondly we achieve better classification results than the high precision weights. This is due to quantization which induces sparsity in the network. One of the well-known remedy for reducing overfitting and improving generalization is "dropout" [11]. However the same work states two facts about dropout effectiveness for CNN. An important consequence of the convolutional shared-filter architecture is a drastic reduction in the number of parameters relative to a neural net in which all neurons apply different filters. This reduces the net's representational capacity, but it also reduces its capacity to overfit, so dropout is far less advantageous in convolutional layers [11]. The proposed work shows that quantizing the network to reduced world length resets 17.3% of network parameters to zero. When we only analyze the convolution layers, 19.2% parameters of the convolution layers are reset to zero. This leads to regularization impact on the network, which results in faster processing and better generalization.

The rest of the paper is organized as follows. Section 2 presents direct quantization and L2 error minimization. Section 3 explains network retraining with quantized weights. Experimental results are provided in Section 4. Finally Section 5 concludes the work.

2. DIRECT QUANTIZATION AND LAYER-WISE SENSITIVITY ANALYSIS

The proposed work divides a network into layer-wise signal groups for quantization [12]. Biases and subsampling layer parameters are kept in high precision. Throughout this article, high precision refers to single precision floating point. Each convolution kernel is treated independently and has its own quantization step size. For example, if the first convolution layer has 1x6 feature maps and 5x5 receptive field, then we have 6 convolutions in total. Each convolution operation shares a single quantization step size among 5x5 = 25 weights. However each fully connected rear end layer has one quantization step size. In uniform quantization, the network is quantized with the same number of quantization levels for all layers. However our sensitivity analysis shows that better savings can be obtained with layer-specific quantization levels.

$$Q(x) = f(z) \quad (1)$$

$$z = g(x) \quad (2)$$

$$f(z) = \Delta . z \quad (3)$$

$$g(x) = sign(x).\min\left\{floor\left(\frac{|x|}{\Delta}+0.5\right), \frac{M-1}{2}\right\} \quad (4)$$

$$E = \frac{1}{2}\sum_{i=1}^{N}(Q(x_i)-x_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\Delta . z_i - x_i)^2 \quad (5)$$

$$Z(t) = argmin_z\ E\left(x, z, \Delta^{(t-1)}\right) \quad (6)$$

$$\Delta^{(t)} = \frac{\sum_{i=1}^{N} x_i z_i^{(t)}}{\sum_{i=1}^{N}(z_i^{(t)})^2} \quad (7)$$

Listing 1. Quantization with L2 error minimization

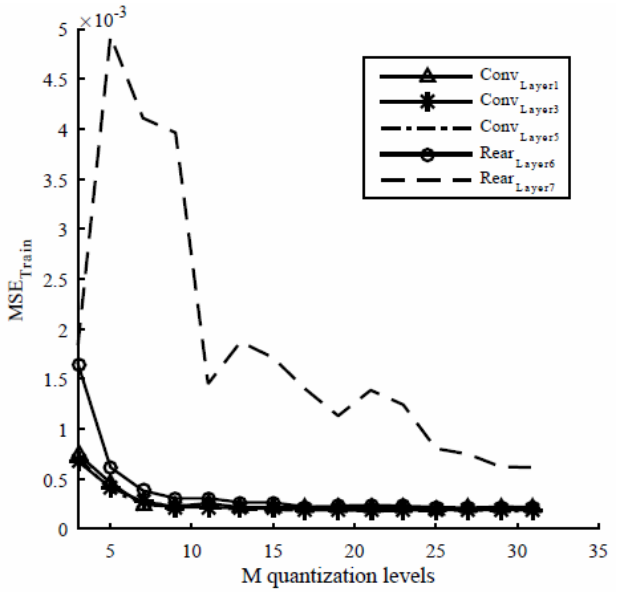

Fig. 2. Layer wise sensitivity analysis. The y-axis represents the network MSE on the training set while the x-axis represents M quantization levels. The plot shows that rear layers are more sensitive to quantization noise than convolution layers.

2.1. L2 error minimization

The L2 error minimization quantizes weights with an optimum quantization step size. The L2 error minimization criterion is the same as reported in [9] and listing 1. The approach is similar to Lloyd-Max quantization except that the quantizer is uniform. *Q(x)* represents the quantization function, Δ shows the quantization step size, *M* represents the number of quantization levels and *z* shows integer membership. The iterative L2 error minimization procedure is outlined in Eq. (1) to (7). This algorithm yields quite

Table 1 Direct quantization results (MNIST)

Convolution Layers (M Levels)			Rear Layers (M Levels)		Test Set MCR (%)
C1	C3	C5	F6	F7	
3	5	5	7	31	1.68
5	5	5	15	31	1.73
7	7	7	15	31	1.10
7	7	7	15	255	1.02

Table 2 Layer wise network parameters distribution pre and post quantization (MNIST)

Layers		Convolution layers			Rear end NN layers		Overall
		C1	C3	C5	F6	F7	
M Quantization Levels		7	7	7	15	31	
Per layer weights count		150	2400	48000	10080	840	61470
Zeroed Weights	Pre Quantization	0	2	47	10	2	61
	Post Quantization	29	496	8477	1346	314	10662
	Percentage (%)	19.3	20.6	17.6	13.3	37.1	17.3

accurate quantization step size for convolution layers. However, for the rear layers, we apply a heuristic search for determining the optimum quantization step size. This search is centered on the initial step size obtained from Eq. (1) to (7). Results with direct quantization are reported in Table 1. The corresponding floating point misclassification rate (MCR) is 0.81%. We can find that miss classification rates with directly quantized network are not good. The network performance can be improved with retraining.

2.2. Layer wise sensitivity analysis for non-uniform quantization

Uniform quantization means applying the same value of M to all layers, $M_{conv} = M_{rear}$. In order to squeeze the full benefits of quantized network, we conduct quantization sensitivity analysis for all convolution and rear layers. The pooling layers are excluded as these are kept in high precision. The sensitivity analysis procedure for a general signal processing system is outlined in [12]. The sensitivity analysis helps in computing the optimum value of M for each layer. We quantize one layer and keep the other layers in high precision. This process is conducted one by one for all the layers. The sensitivity analysis plot is shown in Fig. 2. The analysis is conducted for M = 3, 5… 31. Note that M of 31 corresponds to 5 bits quantization. The mean square error (MSE) on training set is recorded when the network converges. Figure 2 shows that the weights between the penultimate and final layer, $rear_{layer7}$ are most sensitive to quantization. This is due to the addition of quantization noise closer to the network output. In our remaining experiments we keep this layer with 5 or 8 bit precision. $Rear_{layer6}$ is the second most sensitive layer and we keep it with 4-bit precision. Convolution layers are proved to be robust to quantization and are kept in 3 bit precision. The next section discusses the retraining mechanism with quantized weights.

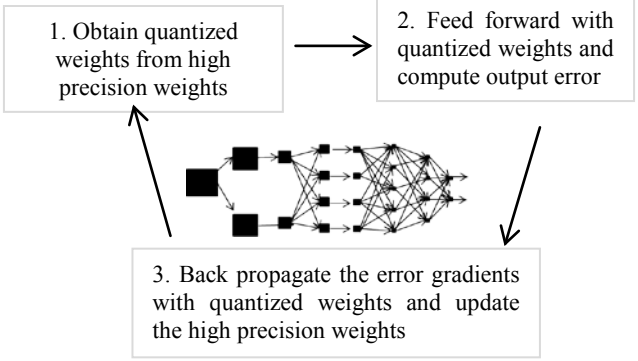

Fig. 3. Network retraining with quantized weights

3. NETWORK RETRAINING

Direct quantization reduces the word length at the cost of degraded performance. Therefore retraining is desired to improve classification. As the theory of error back propagation [13] is well known, we focus here on explaining error back propagation in the context of our quantization framework. During training we keep parameters in both high and low precision. We set aside 5000 training samples for validation. This set is used to decide the network convergence criterion. We start with a high precision pre trained network and obtain a quantized network using L2 error minimization. Then the inputs are fed forward via the network with the low precision weights. This way the output error is indirectly driven by the quantization process. The output error is back propagated via low precision weights. The computed change in weights is added to the high precision weights. Thus we obtain new high precision weights. This process is iterated for several mini-batches and epochs. During training the selection of mini-batch size is important. Generally CNN employs the stochastic gradient descent (SGD) algorithm, where conventionally the mini-batch size is one and weights are updated after each sample. This is quite noisy but helps CNN to avoid trapping in local optimums. Reducing the word length is also noisy due to quantization. Our simulations show that when these two noises add up, convergence is very slow. Therefore the batch size selection is important. Our experiments show that the mini batch size of 50 is a good choice.

Table 2 shows a layer wise distribution of weights after undergoing non-uniform quantization and retraining. This network achieves 0.81% MCR on test set with high precision weights. After retraining with quantization, 17% weights are set to zero. The network achieves 0.84% MCR with low precision weights. Further it is evident that on the average more sparsity is induced in the rear layers compared to convolution layers. Each convolution kernel shares one quantization step size. Due to fewer parameters in the

Table 3 Retrained quantized network (MNIST)

Signal Quantization (bits)	Quantization Levels, M					Test set MCR (%)
	C1	C3	C5	F6	F7	
5	7	7	7	15	15	0.92
	7	7	7	15	31	0.88
	7	7	7	15	255	0.89
8	7	7	7	15	15	0.91
	7	7	7	15	31	0.88
	7	7	7	15	255	0.91
32	7	7	7	15	31	0.84
	7	7	7	15	255	0.77

convolution operation, the induced sparsity is lesser than the rear layers.

4. EXPERIMENTAL RESULTS

Our simulations consist of thousands of epochs of runs and huge networks. It is therefore very time consuming to train the network on CPU. We implemented a highly parallel CUDA based GPU program. Convolution involves accessing each location 25 times for 5x5 masks. We therefore map each convolution operation to a single thread block. This resulted in better utilization of shared memory as a cache. Secondly global memory contents are copied to shared memory in a coalesced fashion. During back propagation of error gradients, we pull the gradients instead of push, which results in better memory access [14]. We obtain two orders of (> 100 times) faster processing compared to single core CPU implementations on big networks (CIFAR-10 network). Our results consider max pooling [3]. We do not conduct experiments with average pooling not only because max pooling is superior to average pooling but also max pooling does not involve bias and weight. The learning rate starts at 0.01 and is decremented after each epoch by multiplying with 0.98.

4.1. Handwritten digit recognition (MNIST)

MNIST is a handwritten digit recognition dataset consisting of 60,000 training and 10,000 test samples. Each sample has 32x32 resolutions and is gray scale. We experiment with CNN architecture having 1-6-6-16-16-120-84-10 layer wise feature maps. We train the network with rectified linear units (ReLUs). Table 3 shows the classification results. We can find that higher precision for the rear layer results in better classification. The quantized network performs better, similar or comparable classification with only 10% memory space consumption. Signal quantization reduces the required hardware but slightly increases the misclassification rate. All results are obtained using 2x2 max pooling.

4.2. CIFAR-10 object recognition

We also evaluate the quantized network on a general object recognition dataset, CIFAR-10 [5]. This dataset consists of ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The training set consists of 50,000 RGB samples. Test set contains 10,000 samples. Each sample has 32x32 resolution. We experiment with the CNN network having 3-64-64-64-64-64-184-10 feature maps. This is a big network and incorporates more than 8K convolution connections. This network architecture has similarity to the one reported in [15]. However our simulations do not consider response normalization layers, overlapped pooling and other optimizations. Before training starts, we preprocess the training samples. The mean activity over the training set is subtracted from each pixel [6]. This is to account for variations in illumination. This is done for each channel. We train the network with rectified linear units and max pooling. The high precision network achieves 33.95% MCR on the test set. The quantized network with $M_{C1/C2/C3} = 7$, $M_{F6} = 31$ and $M_{F7} = 255$ obtains test set MCR of 33.34%.

5. CONCLUSION

This work provides a training mechanism with quantized weights. The resulting network can perform accurate classification with reduced word length. Further the induced sparsity helps the network to generalize well. The proposed work is good for efficient hardware and software implementations.

7. REFERENCES

[1] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology,* vol. 195, no. 1, pp. 215-243, 1968.

[2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of the IEEE,* vol. 86, no. 11, pp. 2278- 2324, 1998.

[3] D. Scherer, A. Muller and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks*, 2010, pp. 92-101, 2010.

[4] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv: 1301.3557,* 2013.

[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.
[6] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. of the Advances in Neural Information Processing Systems,* 2012.
[7] S. L. Phung and A. Bouzerdoum, "MATLAB library for convolutional neural network," Technical Report.
[8] J. Kim, K. Hwang, and W. Sung, "x1000 real-time phoneme recognition, VLSI using feed-forward deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on,* IEEE, pp. 7510-7514, 2014.
[9] K. Hwang and W. Sung, "Fixed-point feed forward deep neural network design using weights +1, 0 and -1", in *Signal Processing Systems (SiPS), 2014 IEEE workshop on, IEEE 2014*.
[10] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. Lecun and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Circuits and Systems (ISCAS), Proc. of IEEE International Symposium on*. IEEE, pp. 257-260, 2010.
[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv: 1207.0580*, 2012.
[12] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *Signal Processing, IEEE Transactions on,* vol. 43, no. 12, pp. 3087–3090, 1995.
[13] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," in *Letters to Nature*, vol. 323, pp. 533-536, 1986.
[14] P. Y. Simard, D. Steinkraus and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Document Analysis and Recognition (ICDAR), 2003 IEEE International Conference on*. IEEE, pp. 958-963, 2003.
[15] A. Krizhevsky. *cuda-convnet* [online]. Available: https://code.google.com/p/cuda-convnet/