LEARNING SEPARABLE FIXED-POINT KERNELS FOR DEEP CONVOLUTIONAL NEURAL NETWORKS

Sajid Anwar, Kyuyeon Hwang and Wonyong Sung

Department of Electrical and Computer Engineering
Seoul National University
Seoul, 08826 Korea
Email: sajid@dsp.snu.ac.kr, khwang@dsp.snu.ac.kr, wysung@snu.ac.kr

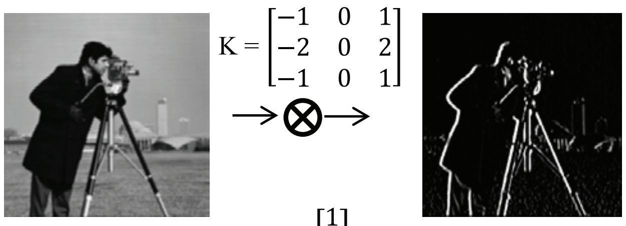
ABSTRACT

Deep convolutional neural networks have shown outstanding performance in several speech and image recognition tasks. However they demand high computational complexity which limits their deployment in resource limited machines. The proposed work lowers the hardware complexity by constraining the learned convolutional kernels to be separable and also reducing the word-length of these kernels and other weights in the fully connected layers. To compensate for the effect of direct quantization, a retraining scheme that includes filter separation and quantization inside of the adaptation procedure is developed in this work. The filter separation reduces the number of parameters and arithmetic operations by 60% for a 5x5 kernel, and the quantization further lowers the precision of storage and arithmetic by more than 80 to 90% when compared to a floating-point algorithm. Experimental results on MNIST and CIFAR-10 datasets are presented.

***Index Terms*—** convolutional neural network, separable kernels, word-length optimization

1. INTRODUCTION

Deep convolutional neural networks (CNN) have shown quite good performance at various computer vision applications [1] [2]. A CNN network mainly consists of three types of layers: convolution, pooling and fully connected layers. Conceptually the convolution layers perform feature extraction while the rear end fully connected layers conduct classification. Usually most of the computations are performed in the convolution layers [1] [2] [3]. Further, bigger sized convolution kernels may have improved representational capacity at the cost of increased processing. It is therefore highly desired to optimize these convolution layers. It was proposed by [4] that a 2D matrix



$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \otimes \rightarrow$$

$$K = v * h \; where \; v = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \; and \; h = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Fig. 1. 2D separable Sobel filter decomposed into two 1D filters. Convolving the input image having WH (width x height) dimensions with a *K x K* convolution kernel requires WHKK multiplications and additions. Separable kernels, however require WH(K+K) multiplications and additions. This results in $speedup = WHKK/(WH(K+K)) = K/2$. Further we only need to save K+K weights instead of *K x K*.

can be approximated with the product of two 1D vectors. The singular value decomposition (SVD) was formally introduced in [5] to approximate a set of filters with linear combinations of a small number of basis functions.

A 3x3 Sobel edge detector example is shown in Fig. 1. For a *K x K* filter, the count of weights is reduced to *K + K* and the speedup is $K^2/2K$. The principal of separability is applied in [6] where two sets of separable and non-separable filters are first learnt. The non-separable filters are then approximated as linear combination of separable filters. Jaderberg et al. performs low rank decomposition not only for filters but also in the channel dimension [7]. In [8], the memory and runtime cost is reduced with sparse connectivity in convolution and fully connected layers. Fixed-point optimization of DNN and CNN is proposed in [9] [10] [11] for reduced memory and runtime costs. However all these kernels are either non-separable or learnt in high precision (32 bit).

The proposed work has two contributions. We first approximate the separable kernels from non-separable ones using SVD. This is followed by quantization with reduced word-lengths for decreasing the computational and VLSI implementation cost. In order to compensate for the

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2015R1A2A1A10056051).

performance loss due to filter separation and weigh quantization, these operations are conducted inside of the retraining process so that the network learns the effect of these transforms and quantization.

The rest of the paper is organized as follows. Section 2 introduces separable kernels and the quantization algorithm. Section 3 explains the retraining algorithm. Experimental results are provided in Section 4. Finally, Section 5 concludes the work.

2. LEARNING SEPARABLE QUANTIZED KERNELS

We first train the network with floating point weights using error back propagation [12] and mini-batch gradient descent. The network has the freedom to learn any kinds of filters and no constraint is imposed on it at this stage. Each convolution kernel is then decomposed into 3 matrices using SVD. Eq. (I) and (II) shows this relationship. Matrix *U* and *V* are orthogonal matrices and *S* is a diagonal matrix containing singular values. The reconstructed matrix *W* will be rank 1 if we only use *S(1, 1)* in Eq. (II).

$$[U, S, V] = SVD(W) \qquad \text{(I)}$$
$$W = U * S * V^T \qquad \text{(II)}$$

The SVD kernels and weights in the fully connected layer are then fixed-point optimized which is explained in the following two subsections. The plots in Fig. 2 shows that separability and quantization significantly reduce the required memory for storing convolutional weights. The kernels are only relevant for convolution layers. It is important to note that SVD computation does not demand any cost at the inference time since it is only computed during training.

2.1. Quantization with L2 error minimization

Components in a digital signal processing system may exhibit varying level of sensitivity to quantization noise. Therefore we use different quantization bits for each layer in

$$Q(x) = f(z) \qquad (1)$$
$$z = g(x) \qquad (2)$$
$$f(z) = \Delta . z \qquad (3)$$
$$g(x) = sign(x). \min\left\{floor\left(\frac{|x|}{\Delta} + 0.5\right), \frac{M-1}{2}\right\} \qquad (4)$$
$$E = \frac{1}{2}\sum_{i=1}^{N}(Q(x_i) - x_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\Delta . z_i - x_i)^2 \qquad (5)$$
$$Z(t) = argmin_z\ E\left(x, z, \Delta^{(t-1)}\right) \qquad (6)$$
$$\Delta^{(t)} = \frac{\sum_{i=1}^{N} x_i z_i^{(t)}}{\sum_{i=1}^{N} (z_i^{(t)})^2} \qquad (7)$$

Listing 1. Quantization with L2 error minimization [9] [10]

Table 1 Fixed-point optimization without re-training (CIFAR-10), floating point MCR = 26.21%

Separable Kernels	Convolution Layers (*M* Levels)			Rear Layers (*M* Levels)		Test Set MCR (%)
	C1	C3	C5	F6	F7	
No	3	3	3	7	31	77.63
	7	7	7	15	31	42.36
	15	15	15	31	63	47.02
	31	31	31	63	63	32.74
Yes	3	3	3	7	31	81.55
	7	7	7	15	31	60.59
	15	15	15	31	63	59.82
	31	31	31	63	63	59.46

accordance to sensitivity analysis [10] [13]. The results drawn in [10] shows that the rear layers of CNN are comparatively more sensitive to quantization noise. Each convolution kernel is treated independently and has its own quantization step size. However each fully connected layer has one quantization step size. Biases are kept in high precision. Throughout this paper high precision refers to 32-bit floating point. Eq. (1) to (7) in Listing 1 outlines the L2 error minimization criterion. The goal is to quantize the input vector with an optimum quantization step size Δ. This criterion uses a uniform quantizer and bears similarity with the Lloyd-Max quantization [9]. *Q(x)* represents the quantization function, *M* represents the number of quantization levels and *z* shows integer membership. When *M* is 7, it demands 3 bits for weight representation. The obtained quantized network shows degraded performance compared to the floating point network.

2.1. Directly quantizing the separable kernels

Approximating a non-separable matrix with a rank1 matrix

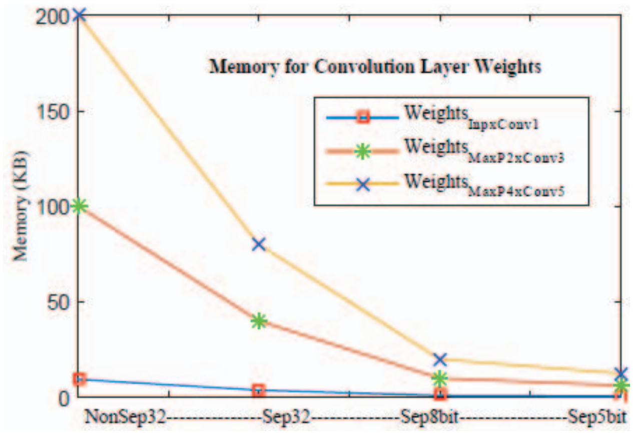


Fig. 2. Memory required for convolution layers weights of CIFAR-10 network. For *K*×*K* kernel, separability causes *K/2* reduction. The quantization further reduces the storage by 4 (8bits) to 6(5bits) times. The 3rd convolution layer employs 32x64 convolution connections. The required memory storage is reduced from 200 KB (32x64x5x5x32) to 12.5-20 KB with quantized separable kernels.

Separable quantized kernels (convolution layers):

$$W = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$

$$[U, S, V] = SVD(W)$$
$$v = U(:,1) * \sqrt{S(1,1)}$$
$$h = V(:,1)^T * \sqrt{S(1,1)}$$
$$W^{(s)} = v * h$$
$$\text{Update: } W = W^{(s)}$$
$$W^{(sq)} = Q_w(v) * Q_w(h)$$

Quantized fully connected layer:

$$w_{ij}^{(q)} = Q_w(w_{ij})$$

Forward activation:

$$net_i = \sum_{j \in A_i} w_{ij}^{(q)} y_j^{(q)}$$
$$y_i^{(q)} = Q_s(f(net_i))$$

Backward step:

$$\delta_j = f'(net_j) \sum_{i \in Pj} \delta_i w_{ij}^{(q)}$$

Output error calculation:

$$\frac{\partial E}{\partial w_{ij}} = -\delta_i y_j^{(q)}$$

Weight update:

$$w_{ij,new} = w_{ij} - \alpha \left\langle \frac{\partial E}{\partial w_{ij}} \right\rangle$$

Fig. 3. Network retraining with quantized weights and separable kernels. The non-separable version of this algorithm is reported in [9] [10]. *W* shows single precision *k*×*k* kernel (*3×3* here). $W^{(sq)}$ is a separable quantized kernel, *E* is the output error, $Q_s(.)$ and $Q_w(.)$ represents the signal and weight quantizer respectively, *f(.)* is the activation function, δ_i is the error signal of unit *i*.

introduces distortion. Secondly the word-length optimization is also noisy. These two noises when add up, degrades the network performance severely. Table 1 shows the network performance with directly quantized weights and kernels. C1, C3 and C5 represent convolution layers whereas F6 and F7 denote the rear end fully connected layers. We can find that direct quantization degrades the network performance. When the separability constraint is also imposed on the quantized kernels, the performance further degrades severely and the misclassification rate (MCR) shoots up by a bigger margin. The corresponding floating point MCR is much lower compared to the directly quantized separable network. This shows that re training the network is highly desired to compensate for these losses.

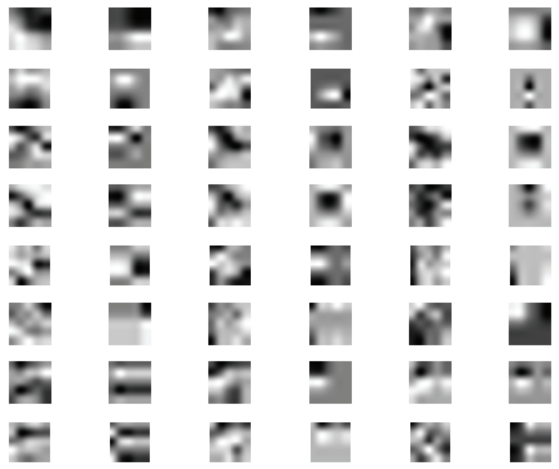


Fig. 4. The 5x5 high precision kernels are shown in the odd columns (1, 3, 5). The even columns show their corresponding separable and quantized (M=7) approximations.

3. NETWORK RETRAINING

We re-train the directly quantized network with our algorithm outlined in Fig. 3. We modify the error back propagation algorithm [12] in such a way that we can learn separable quantized kernels. During retraining we maintain both high and low precision weights. We initialize the network using the pre-trained floating point network. Each high precision 2D convolution kernel *W* is decomposed into two 1D vertical and horizontal filters (*v, h*) using SVD. At this stage both *v* and *h* are in high precision. We reconstruct the rank 1 high precision $W^{(s)} = v*h$. The difference between *W* and $W^{(s)}$ ($W - W^{(s)}$) depends on the linear independence of columns or row vectors of *W*. In our analysis, smoother error gradients are obtained when we update *W* with $W^{(s)}$ ($W = W^{(s)}$). The *v* and *h* vectors are then quantized with *M* quantization levels and we obtain $v^{(q)}$ and $h^{(q)}$. These two 1D vectors now represent the 2D separable quantized kernel $W^{(sq)}$. The rear end fully connected layer weights are also quantized using L2 error minimization. These quantized kernels and weights are then used in the forward path of the network. This way the network output (and the network error = *t* - *y*) is driven by the quantization and separability constraints. The low precision weights and separable quantized kernels are used to propagate the output error backwards. The error is accumulated in high precision and floating point weights are updated. The newly obtained high precision weights are again made separable and quantized in the next iteration.

4. EXPERIMENTAL RESULTS

We evaluate our proposed algorithm with two examples: handwritten digit recognition (MNIST) and a ten class general object recognition (CIFAR-10). We provide

Table 2 CIFAR-10 retraining with fixed-point separable kernels, $MCR_{non\text{-}sep\text{-}real}$ = 26.21% and $MCR_{sep\text{-}real}$ = (26.76) %

Convolution Layers (*M* Levels)			Rear Layers (*M* Levels)		Test Set MCR (%)
C1	C3	C5	F6	F7	
3	3	3	7	31	33.05
7	7	7	15	31	27.58
15	15	15	31	63	26.73
31	31	31	63	63	26.88
63	63	63	63	63	26.79

Table 3 MNIST retraining with fixed-point separable kernels, MCRnon-sep-real = 0.91% and MCRsep-real = 1.08%

Convolution Layers (*M* Levels)			Rear Layers (*M* Levels)		Test Set MCR (%)
C1	C3	C5	F6	F7	
3	3	3	7	31	1.12
7	7	7	15	31	0.94
15	15	15	31	63	1.03
31	31	31	63	63	1.01
63	63	63	63	63	1.08

misclassification results with three cases: high precision and non-separable, high precision and separable, quantized precision and separable. Fig. 4 compares some of the non-separable high precision kernels with the re-trained quantized precision separable kernels. The separable quantized kernel shapes (even columns) exhibit the linear dependence between row or column vectors in a rank one matrix. During training, we used a mini batch size of 128 and RMS prop [14]. Rectified linear units (ReLUs) are used as activation functions. For convergence and learning rate annealing, we set aside a subset of training samples as the validation set for both experiments.

4.1. CIFAR-10 object recognition

The CIFAR-10 dataset consists of ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck [15]. The training set consists of 50,000 samples. Test set contains 10,000 samples. Each sample has 32x32 RGB resolution. We conduct the experiment with the CNN network having 3-32-32-32-32-64-64-10 feature maps. This network architecture is reported in [16]. All the three convolution layers use 5x5 kernels. The first subsampling layer employs max pooling while the second one uses average pooling. The results are reported in Table 2. The high precision network has MCR of 26.21% on the test set with overlapped pooling. The floating point precision network achieves MCR of 26.76% with separable kernels after retraining. Results in Table 2 shows that the resource efficient quantized networks achieve comparable performance with the floating point network.

4.2. Handwritten digit recognition (MNIST)

MNIST is a handwritten digit recognition dataset consisting of 60,000 training and 10,000 test samples [1]. Each sample is resized to 32x32 gray scale resolution. We conduct experiment with CNN architecture having 1-6-6-16-16-120-84-10 layer-wise feature maps and rectified linear Units (ReLUs). Table 3 shows the classification results. The high precision separable network achieves MCR of 1.08% at convergence. The quantized precision separable network performs slightly better. To evaluate the effect of signal quantization on separable quantized weights and kernels, the hidden layers are quantized with 5 bits and the output layer is kept in high precision. The resulting network achieves MCR of 0.95%. These results show little effect of signal quantization and are consistent with other findings [9] [10]. The separable and fixed-point optimized network performs comparable classification with only 6.5% memory space consumption.

5. CONCLUSION

Convolution layers that incur higher computational complexity are hot spots for optimizations. The separability constraint greatly reduces the computational cost and memory requirement. The constraint-less initial learning with floating point weights enables the network to achieve baseline performance while the late introduction of separability constraint and re-training ensures resource efficiency. The fixed-point optimization further reduces the word-length and results in power efficiency which may be crucial for embedded systems. Our technique is generic and is useful for both software and hardware implementations.

7. REFERENCES

[1] 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.
[2] 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.
[3] S. Karen and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[4] S. Treitel and J. Shanks, "The design of multistage separable planar filters," in *IEEE Transactions on Geoscience Electronics*, vol. 9, no. 1, pp. 10-27, 1971.
[5] P. Perona, "Deformable kernels for early vision," in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 17, no. 5, pp. 488-499, 1995.
[6] A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit and P. Fua, "Learning separable filters," in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 1, pp. 94-106, 2015.

[7] M. Jaderberg, A. Vedaldi and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” arXiv preprint arXiv:1405.3866 (2014).

[8] M. Collins and P. Kohli, “Memory bounded deep convolutional networks,” *arXiv preprint arXiv:1412.1442*, 2012.

[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] S. Anwar and W. Sung, “Fixed-point optimization of deep convolutional neural for object recognition,” in *Proc. of the IEEE international conference on acoustics, speech, and signal processing (ICASSP)*, 2015.

[11] K. Jonghong, K. Hwang, and W. Sung, "X1000 real-time phoneme recognition VLSI using feed-forward deep neural networks." in *Proc. of the IEEE international conference on acoustics, speech, and signal processing (ICASSP)*, 2014.

[12] 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.

[13] 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.

[14] G.E. Hinton, N Srivastava and K. Swersky, “Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude,” 2012.

[15] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Master’s thesis, Department of Computer Science, University of Toronto, 2009.

[16] A. Krizhevsky. *Cuda-convnet* [online]. Available: https://code.google.com/p/cuda-convnet/