Performance Analysis of Fixed Point Bit Precision for Output Fmaps in CNNs

Anaam Ansari*†, Tokunbo Ogunfunmi*‡
*Department of Electrical Engineering
Santa Clara University, Santa Clara, California 95053

***Abstract*—Image classification and speech processing have made Convolutional Neural Networks (CNN) mainstream. Due to their success, many deep networks have been developed such as AlexNet, VGGNet, GoogleNet, ResidualNet [1]–[4],etc. Implementing these deep and complex networks in hardware is a challenge. There have been many hardware and algorithmic solutions to improve the throughput, latency and accuracy. There are many model compression techniques that have been introduced to decrease the model size and help develop a generalized model. Quantization is one such technique. Traditionally, quantized weights and inputs are used to reduce the memory transfer and power consumption due to it. In this paper, we use quantization on the output of each layer for AlexNet and VGGNET16 and analyze its effect on the soft-max score during forward pass given the prediction is correct. We found that although, there is a deterioration in the softmax confidence score we can lower the bit precision of the output fmap data before suffering a considerable loss. AlexNet performs similarly with single (32 bit) floating point precision and a signed 12 bit fixed point number whereas VGGNET16 using double floating point precision (64 bits) and signed 24 bit fixed point number give a comparable performance.**

***Index Terms*—CNN, compressed network, AlexNet, quantization, fixed point.**

I. Introduction

Convolutional Neural Networks are comprised of multiple layers that have different functions. These layers take an input volume and transform it into an output volume. Successive processing of these layers results in weights that are updated heuristically with each forward pass of input to the output. The operations performed on the input, in a convolutional layer are described in the figure 1 [5]. The stages in the covolutional layer are the input stage, the convolutional stage, the non linear stage and the pooling stage.

The input stage prepares the input volume to be processed by the convolutional stage by conforming it to prescribed dimensions. The prepped input is given to the convolutional stage. This is the stage that does most of the heavy lifting in terms of computational cost such as matrix multiplications and vector multiplications. The convolutional operation is described in algorithm 1. CNNs are neural network that are sparsely connected as they rely on the spatial properties of the input payload. Conventionally, this is the stage where the quantized weights are used. We want to perform fixed point arithmetic and quantize the output

†aaansari@scu.edu
‡togunfunmi@scu.edu

There is a nonlinear stage to map the output of the convolutional stage to a continuous limited range. Activation functions are supposed to emulate the generation of 'action potential' in a neuron [6]. The purpose of the the activation function is to provide a smooth unique mapping. Some of the popular functions used in the non-linear stage are $tanh()$, $sigmoid$, $\sigma()$ and $relu, max(0, k)$. They are described in the figure 3. The non linear function $relu$ is used in AlexNet and VGGNET16 as it limits the output to a range $(0, 1)$ [1].

The maxpooling stage condenses the result of the nonlinear stage into a smaller size. This is akin to down sampling that removes redundant data and makes the processed volume manageable while conserving all the key features and information in it. Pooling converts the output of the net at a certain location to statistics of neighboring outputs. The pooling layer output is invariant to any small translations in the input. This makes the presence of a feature take precedence over th location of the feature [7]

This is where we propose the new quantization scheme to lie. We want to assess the quality of performance of the device by quantizing the output of the convolutional layers.

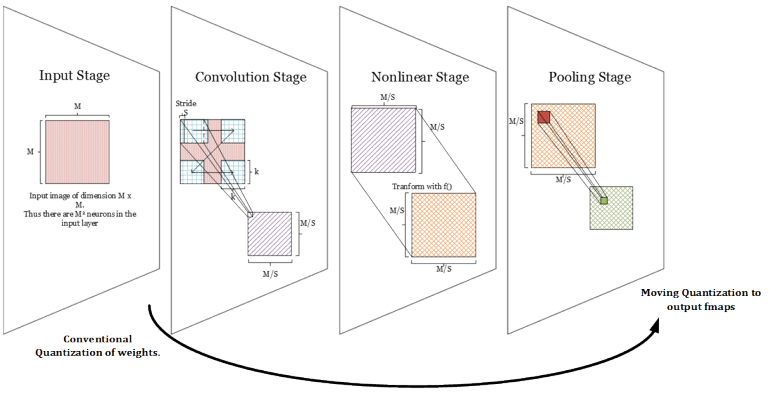

Fig. 1. The functions involved in computing a CNN layer

A CNN architecture can be organized to have many layers in many combination. The engineering decisions about the depth and size of model can be used to optimize the latency, throughput and accuracy of the network.

II. Motivation

Techniques like dropout and pruning have been introduced to address problems like overfitting [8], [9]. They have also

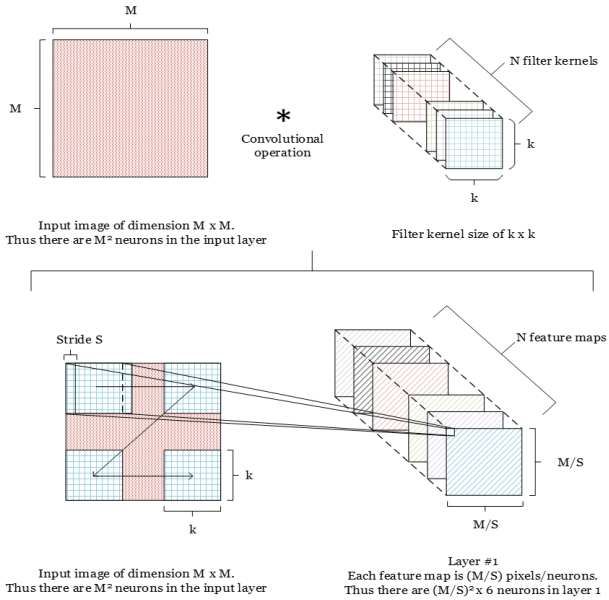

Fig. 2. convolutional operation

input : M inputs, $Imap$ of size $Win \times Hin$
input : Input weights $kernels$ of size $k \times k$
The stride of the kernels is S
output: $Ofmap$ of size $Wout \times Hout$

for $(w = 0, w < Wout, w++)$ **do**
  for $(h = 0, h < Hout, h++)$ **do**
    for $(n = 0, n < N, n++)$ **do**
      $wStart = (w-1) * S + 1;$
      $wEnd = wStart + k - 1;$
      $hStart = (h-1) * S + 1;$
      $hEnd = hStart + k - 1;$
      $Ofmaps(w,h,n) =$
      $Ofmaps(w,h,n) +$
      `Accumulate(`
      $Imap(wStart : wEnd, hStart : hEnd, 1 : M)$
      $*kernels(:,:,1 : M, n))$
)
    end
  end
end

Algorithm 1: 2D convolutional operation

helped to compress model sizes. Weight pruning and quantization techniques are used in many hardware devices. [10]–[14] Works like **DeepCompression** [15] have proved that a combination of network optimization techniques such as pruning, quantization and Huffman coding prove very successful in reducing the network size while maintaining accuracy as can be seen in Table I. Most of these techniques are applied to weights since, dense networks are really large and have a large number of parameters. For example AlexNet has about 60M parameters [1] and VGGNET16 has 138M [4]. This strategy is valid and will reduce the memory transfer between

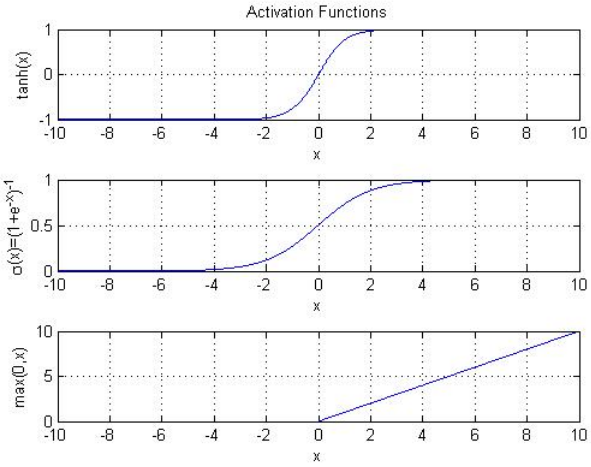

Fig. 3. Non-linear activation functions

computation engine and the main memory.

In this paper, we want to explore the effect of quantization in between convolutional layers. We observe the effect of various fixed point bit precisions in the forward path of a network. We examine the results that correctly predict the test image and quantify the **softmax** score corresponding to a bit precision. A softmax classifier is a generalization of the Logistic Regression Classifier. It defines multiple output classes compared to binary output classes. It takes the input vector and converts it into a normalized vector between 0 and 1 and the sum of that vector is also equal to 1.

CNN layers are most often processed iteratively. Thus, the outputs of the CNN layers need to be transported back to memory. As a result, we are interested in reducing the size of the output fmaps. This will help us reduce the memory read and write cycle. Memory transfer for voluminous calculations for convolutional layers and fully-connected layers contributes to a substantial energy consumption and hence power requirement to run the hardware. There are many solutions that address this problem with data reuse and datapath design [16]–[19]. Reducing the amount of output fmaps that need to be written back to memory and read again for the next layer will be helpful in memory transfer reduction. Our work will help contribute to this reduction of memory traffic by quantizing output fmaps into fixed point bit precision

III. Networks

The year 2012 was a momentous year for Deep Learning. The very first Deep Learning entry to the ImageNet challenge [20] in 2012 was AlexNet [1]. It had 60M parameter and was implemented on GPUs. In this paper, we focus on AlexNet (1st place winner) and VGGNET16 (1st runner up) for the analysis. The analysis can similarly be done for other networks as well.

A. AlexNet

The overall architecture of AlexNet has 8 layers as seen in figure 4. The first five layers are convolutional layers. They

TABLE I
EFFECT OF QUANTIZATION OPTIMIZATION TECHNIQUES [12], [15]

Network	Orignial Size (MB)	Compressed Size (MB)	Compression Ratio	Original Accuracy	Compressed Accuracy
AlexNet	240	6.9	35x	80.27%	80.30%
VGGNet	550	11.3	49x	88.68%	89.09%
GooglNet	28	2.8	10x	88.90%	88.92%
SqueezeNet	4.8	0.47	10x	80.32%	80.35%

perform convolutional operations as described in figure 2. The output of the network is the output of the fully connected layer which is subjected to a softmax probability based classification which can classify an input into 1000 classes. The input to the network is an image of size $227 \times 227 \times 3$ with a fixed stride. The stride is 4 pixels for AlexNet. This is operated on by 96 filter kernels of size $11 \times 11 \times 3$. The output thus obtained is put through a non-linear $relu$ activation function and maxpooled. The output of the first convolutional layer are 96 fmaps of size 55×55. This is acted upon by the second convolutional layer which has 256 kernels of size $5 \times 5 \times 48$. The third, fourth and fifth convolutional layers act similarly and their kernel sizes and number are described in Table II. The fully connected layers (fc1) flattens the dense convolutional layer into dimensions 4096×9216. The fc2 and fc3 fully connected layers culminate the network by distilling the output of the complicated network into 1000 statistically learned classes.

TABLE II
FILTER KERNELS FOR EACH LAYER

layer	height	width	channels	kernels
conv1	11	11	3	96
conv2	5	5	48	256
conv3	3	3	256	384
conv4	3	3	192	384
conv5	3	3	192	256
fc1	4096	9216	1	1
fc2	4096	4096	1	1
fc3	1000	4096	1	1

B. VGGNET16

The VGGNET16 has 16 weight layers - 13 convolutional layers and 3 fully connected layers. Instead of using variable sized kernel like AlexNet, VGGNET16 uses a uniform kernel size 3×3. They use this small sized receptive field through out the network. The stride and padding are 1 pixel each. [4]

IV. FIXED POINT ARITHMETIC ANALYSIS

Fixed point arithmetic is best suited for most hardware applications because of low cost. Thus, we are exploring various fixed point precisions to reduce the output fmap read and write cycles. We use the following notation $fixed < sM, N >$ to represent the bit precision of the fixed point number. S represents signed, M is the complete word length and N is the fractional word length . We use fixed point precision analysis as it would help us carry out precision sensitive arithmetic on the input payload. Manipulating the fractional length along with the word length also helps us investigate the network's sensitivity to integer length and fractional length.

In our analysis, we have used the AlexNet network forward pass that was created by [21], [22] and modified it for fixed point operation. We use the same network deconstruction for VGGNET16 forward pass. We transform all the inputs and outputs to fixed point formats and run the following experiments.

- AlexNet forward pass
  - baseline single precision forward pass
  - Input precision s32,22, layer precisions [s32,22 s26,7 s16,7 s12,4 s12,5 s12,6 s10,4 s8,2 s6,2] output precision double
  - Input precision s16,7, layer precisions [s16,7 s12,4 s10,4 s8,2 s6,2] output precision double
  - Input precision s8,2, layer precisions [s8,2 s6,2] output precision double
- VGGNET16 forward pass
  - baseline double precision forward pass
  - Input precision s45,17, layer precisions [s32,22 s24,10 s16,6] output precision double
  - Input precision s16,7, layer precisions [s24,10 s16,7] output precision double

Analyzing preliminary runs of the forward pass gives us some clue about the bit precision that could accommodate the range of the output fmaps. The experiments we run, test the limit of the network's robustness against inadequate representation for output fmap values while also exploring the underlying loss that we can afford. Single bit precision is adequate to represent AlexNet weights and double precision is adequate to represent VGGNET16 weights. These precisions are used as baselines for both networks.

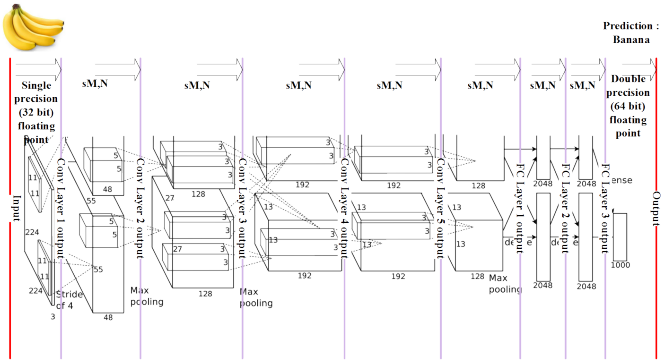

Fig. 4. Forward Path of AlexNet with fixed precision sM,N

V. Results

The Tables III,IV,V and VI show in detail the results of our analysis. They describe the prediction status (TRUE/FALSE) and the softmax score. Tables III and IV show the result from AlexNet and Tables V and VI show the results from VGGNET16. In both sets of results we can see the deterioration of the softmax score as the bit precision goes down.

In Table III we compare the results achieved using fixed point precision with a single precision (32 bit) floating point. For this setup the input and filter weights are of s32,22 fixed point precision. We can see here that the forward pass of the network is able to correctly identify the input image of a banana however, its confidence scores keeps deteriorating with the smaller precisions. We see a good perfromance up until fixed point precision s12,4. The softmax score seems be very large for a correct predicion for s12,4. Changing the fractional length of the fixed point also has an effect on the softmax score as s12,5 performs poorly compared to s12,4. The softmax score keeps declining after s12,4 and we make a wrong prediction at s6,2. In Table IV the input and the weights are s16,7 and we compare various fixed point precisions with the single precision performance. Here the softmax score is high and begins deteriorating after s12,4.

TABLE III
AlexNET:Results for fixed point precision quantization on output fmaps input and weights (s32,22), output double precision

Precision	Prediction (True/False)	softmax score
single	TRUE	9.80E-01
s32,22	TRUE	9.82E-01
s26,7	TRUE	9.82E-01
s16,7	TRUE	9.82E-01
s12,4	TRUE	9.97E-01
s12,5	TRUE	6.68E-01
s12,6	TRUE	1.01E-01
s10,4	TRUE	1.01E-01
s8,2	TRUE	1.00E-01
s6,2	FALSE	2.77E-01

TABLE IV
AlexNET:Results for fixed point precision quantization on output fmaps input and weights (s16,7), output double precision

Precision	Prediction (True/False)	softmax score
single	TRUE	9.80E-01
s16,7	TRUE	9.93E-01
s12,4	TRUE	9.87E-01
s10,4	TRUE	1.01E-01
s8,2	TRUE	1.20E-01
s6,2	FALSE	2.57E-02

In Table V, the input and filter weights are s45,17 fixed point precision. The result of all the fixed point precisions are compared against a double precision performance. Table VI has input and filter weights set to s24,10 precision. It shows similar result from fixed point precisions compared against double precision. The softmax score in both Tables V and VI deteriorates after s12,4 precision.

TABLE V
VGGNET16:Results for fixed point precision quantization on output fmaps input and weights (s45,7), output double precision

Precision	Prediction (True/False)	softmax score
double	TRUE	9.57E-01
s45,17	TRUE	9.57E-01
s32,12	TRUE	9.57E-01
s24,10	TRUE	9.44E-01
s16,6	FALSE	1.26E-01

TABLE VI
VGGNET16:Results for fixed point precision quantization on output fmaps input and weights (s24,10), output double precision

Precision	Prediction (True/False)	softmax score
double	TRUE	9.57E-01
s24,10	TRUE	9.59E-01
s16,6	FALSE	1.23E+00

VI. Conclusion

In this paper we learned that fixed point casting and arithmetic applied to output fmaps has an affect on the softmax score of the CNN. Thus, it will affect the accuracy of the output in test runs. However, we can afford to use lower precision before we see any substantial reduction in the softmax score. We have learned that the softmax score is also sensitive to the word length and fractional length of the output fmaps. Hence we need to pick them with care. It has confirmed our hypothesis and we can now choose the correct precision which can be substantially lower than the input and weight precisions. The performance of a double precision AlexNet forward pass is comparable to a s12,4 fixed point Alexnet forward pass. Similarly the performance of a double precision floating point VGGNET16 forward pass is comparable to a s12,4 fixed point VGGNET16 forward pass. This analysis was done on dense networks and may lead to more saving on sparse networks. This work can be extended to other networks such as ResidualNet , GoogleNet. We can also design a tool that will recommend the best fixed point precision. This technique can be combined with other techniques such as data reuse to design better hardware.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] A. Karpathy. Cs231n: Convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.github.io/

[6] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[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," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 243–254.

[11] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 1–12.

[12] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[13] L. Jiang, M. Kim, W. Wen, and D. Wang, "Xnor-pop: A processing-in-memory architecture for binary convolutional neural networks in wide-io2 drams," in *Low Power Electronics and Design (ISLPED, 2017 IEEE/ACM International Symposium on*. IEEE, 2017, pp. 1–6.

[14] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on rram," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*. IEEE, 2017, pp. 782–787.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[16] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *IEEE Micro*, pp. 1–1, 2017.

[17] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

[18] Y. Shen, M. Ferdman, and P. Milder, "Escher: A cnn accelerator with flexible buffering to minimize off-chip transfer," in *Field-Programmable Custom Computing Machines (FCCM), 2017 IEEE 25th Annual International Symposium on*. IEEE, 2017, pp. 93–100.

[19] ——, "Maximizing cnn accelerator efficiency through resource partitioning," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 535–547.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[21] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design space exploration of fpga-based deep convolutional neural networks," in *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 2016, pp. 575–580.

[22] http://lepsucd.com/. Alexnet forward path. [Online]. Available: https://github.com/pmgysel/alexnet-forwardpath