Evaluating Quantized Convolutional Neural Networks for Embedded Systems

Simon O'Keeffe, Rudi Villing

Department of Electronic Engineering, Maynooth University-National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland

Abstract

This paper presents a deep learning approach which evaluates accuracy and inference time speedups in deep convolutional neural networks under various network quantizations. Quantized networks can result in much faster inference time allowing them to be deployed in real time on an embedded system such as a robot. We evaluate networks with activations quantized to 1, 2, 4, and 8-bits and binary weights. We found that network quantization can yield a significant speedup for a small drop in classification accuracy. Specifically, modifying one of our networks to use an 8-bit quantized input layer and 2-bit activations in hidden layers, we calculate a theoretical 9.9× speedup in exchange for an F_1 score decrease of just 3.4% relative to a full precision implementation. Higher speedups are obtainable by designing a network architecture containing a smaller proportion of the total multiplications within the input layer.

Keywords: Convolutional Neural Networks, Deep Learning, Network Quantization

1 Introduction

Deep Convolutional Neural Networks (CNNs) are recognized as the state of the art for image classification [Krizhevsky et al., 2012]. However, Deep CNNs are too computationally intensive to run on low power embedded devices such as the NAO robot. Quantization of the weights and activations can reduce the computational requirements of Deep CNNs and speed up the inference times. Our previous work [O'Keeffe and Villing, 2017] evaluated various network architectures for ball detection in robot soccer on the Softbank NAO robot using both full precision and binarized networks. Our contribution in this paper is to evaluate the performance of networks with quantized activations and weights binarized to +1 or -1. We also examine the maximum theoretical speedup obtainable for these networks.

2 Related Work

Most of the time taken to perform inference in Deep CNNs results from the multiplication of real-valued weights by real-valued activation values. High precision parameters have been shown to be unimportant to the performance of deep networks [Gong et al., 2014]. As such, network quantization has been proposed to improve the computational efficiency of the network, particularly at inference time. BinaryConnect [Courbariaux et al., 2015] trains a DNN with binary weights during forward and backward propagation, but retains the precision of the stored weights in which gradients are accumulated. The authors found that BinaryConnect acted as a regularizer and obtained near state-of-the-art results on MNIST, CIFAR-10, and SVHN datasets. BinaryNet [Courbariaux and Bengio, 2016] was proposed as an extension of BinaryConnect where both weights and activations are constrained to +1 or -1. If all operands of the convolution are binary, then the convolutions can be calculated by XNOR and bit counting operations. XNOR-Net [Rastegari et al., 2016] is another method that binarizes the weights and activations in a network but uses a different binarization method and network structure than BinaryNet. BinaryNet achieved a top-1 accuracy 27.9% while XNOR-Net achieved 44.2% on the ImageNet 1000 classification task. Both BinaryNet and XNOR-Net are binarized versions of AlexNet [Krizhevsky et al., 2012] which has a 56.6% top-1 full precision accuracy.

In an effort to increase the top-1 accuarcy towards that of a full precision network, several researchers have looked towards using low bitwidth quantization of the acitivations with binary weights. DoReFa-Net [Zhou et al., 2016] trained a selection of networks with low bitwidth activation including a network with 2-bit activations and binary weights achieving a top-1 score of 50.7%. Quantized Neural Networks [Hubara et al., 2016] extended the work on BinaryNet and achieved a top-1 score of 51.03% with 2-bit activations and binary weights. In this paper we examine how the activation bitwidth affects accuracy for our ball detection task and explore the maximum theoretical speedup attainable for various activation bitwidths.

3 Approach and Experiment

Binarizing the weights and activations of a network allows for full precision convolutions to be approximated by XNOR and popcount operations. This can be extended to B-bit quantization with binary weights which increases the amount of time needed to run the XNOR and popcount operation by B. Equation (1) shows the convolution operation with B bit activations and binary weights.

$$s = \sum_{b=1}^{B} 2^{b-1} \left(x^b \cdot w \right) \tag{1}$$

where $x = \{x_1, x_2, \cdots, x_N\}$ is a vector of B bit inputs, x_n^b is the b^th significant bit of the n^th input, w is a vector of 1-bit weights, and s is the resulting weighted sum. Here $\cdot$ indicated the XNOR and popcount operation. The networks were quantized to k-bits according to Equation (2) where $[\cdot]$ indicates the rounding operation and $x \in [-1,1]$

$$q_k(x) = 2\left(\frac{\left[\left(2^k - 1\right) \left(\frac{x+1}{2}\right) \right]}{2^k - 1} - \frac{1}{2} \right) \tag{2}$$

3.1 Maximum Theoretical Speedup

A typical convolutional kernel consists of cN_wN_o multiplications where c is the number of channels, N_w is the number of weights in the kernel, and N_o is the number of outputs resulting from the convolution. When weights are binarized and activations are quantized then a convolutional kernel consists of cN_wN_oB binary operations and N_ob popcount operations, where B is the activation bitwidth. Therefore the maximum theoretical speedup for a convolution relative to full precision on a CPU only can be computed by

$$s = \frac{cN_wN_o}{\frac{1}{R}cN_wN_oB + pN_oB} = \frac{RcN_w}{cN_wb + RpB} \tag{3}$$

where R is the number of binary operations that can be performed in one cycle and p is the number of cycles needed to perform a popcount. Single Instruction Multiple Data (SIMD) operations perform the same operation on multiple data points simultaneously. The Softbank NAO robot has an Intel Atom processor which supports the Streaming SIMD Extensions (SSE) instruction set. Using SSE instructions 128 binary operations ($R = 128$) can be performed in one cycle. The popcount operation can be completed in one cycle on CPUs that support SSE4 instructions, however the NAO robot's processor only supports up to SSSE3. Therefore we assume the popcount operation then takes 4 cycles to perform on average. Equation (3) does not take into account the cost of loading up the SSE registers or storing the result.

Using binary weights eliminates all multiplication operations and reduces convolutions to only additions and subtractions. This can be performed by either using Equation (1) or by packing 8-bit quantized values into 16-bit integers (to prevent overflow) in an SSE register and performing 8 addition operations in parallel. This offers a maximum theoretical speedup of $8\times$.

3.2 Experiment

For our experiment we trained two different network architectures to detect black and white balls in robot soccer environments in the presence of distractors such as field lines and robots. We trained on the Caffe framework [Jia et al., 2014] using a dataset that consists of 16133 20×20 image patches. The full details of the dataset creation can be found at [O'Keeffe and Villing, 2017]. Network 1 consisted of 2 convolution layers with 3×3 kernels and 2 fully connected layers and network 2 consists of 5 convolutional layers using both 3×3 and 1×1 kernels. The networks were trained under two different convolutional block layouts. The first version used the convolutional block outlined by XNOR-Net [Rastegari et al., 2016] and consisted of Batch Normalization, activation, convolution, and pooling layers. The second version followed the typical block in a CNN and consisted of convolution, Batch Normalization, activation, and pooling. Each network was trained in full precision along with networks with 1, 2, 4, and 8-bit quantizations of the activation. Each network was trained for 10,000 iterations.

4 Results

The classification accuracy of the networks can be seen in Figure 1(a). The classification accuracy is given by the F_1 score which is a balanced score between the precision and recall of a given network. We can see that the classification accuracy performs poorly for 1-bit activations, with a mean F_1 score of 85.1%. Moving to 2-bit activations increases the mean F_1 score to 93.6% before increasing to 95.5% and 96.26% for 4 and 8-bit activations respectively.

Since the theoretical speedup only depends on the kernel size and the number of channels, an input image with one channel yields a relatively small speedup as seen in Figure 1(b). The internal layers however are better suited to a speedup from quantization. For example, network 1 with 2-bit activatons yields a 12× speedup. However, internal layers in networks with 4-bit or 8-bit activations do not attain a speedup over SSE-enabled 8-bit integer additions with binary weights.

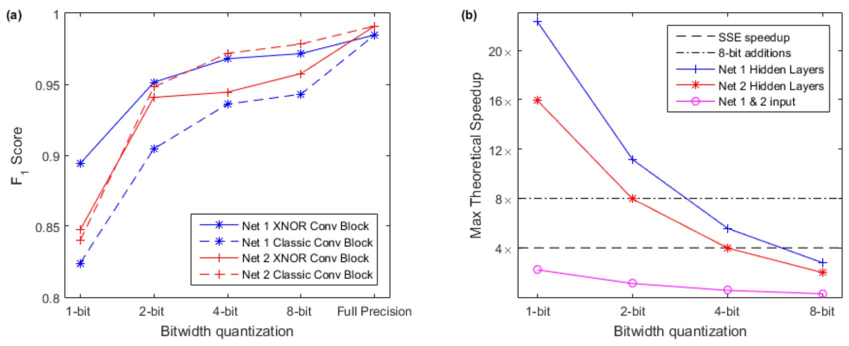

Figure 1: This figure shows (a) the F_1 score classification against the activation bitwidth for both networks and convolution block layouts and (b) the maximum theoretical speedup using XNOR and popcount operations for activation bitwidth

5 Discussion and Conclusion

This evaluation shows that network quantization can yield a significant speedup for a small drop in classification accuracy. For example network 1 with binarized weights and 2-bit activations decreases 3.4% in F_1 score relative to full precision with a potential 12× speedup for the hidden layers. However the theoretical speedup does not hold for the whole network and varies depending on the layer architecture within a network. Layers with a smaller number of channels, such as input layers, do not yield much speedup through XNOR and popcount operations. In these cases full precision convolutions can be implemented using SSE to deliver a 4× speedup relative to CPU only. Alternatively, in exchange for a minor decrease in classification accuracy, an even better speedup of 8× is possible using quantized 8-bit values and the addition technique for convolutions described in Section 3.1.

Convolution speedup in the input layer is not as important an issue for large Deep CNNs such as AlexNet where the input layer has far fewer channels (e.g. 3) than the internal representation (e.g. 512) meaning that the input layer contains only 14.9% of the multiplications for the network. However the ball detection networks evaluated in this work do not use such Deep CNNs, as such the input layer contains 33.4% of the total multiplications. For our networks, we can achieve a 9.9× maximum theoretical speedup for the whole network by using the addition technique for the input layer and 2-bit activations for the hidden layers. Higher speedups can be obtained by designing a network architecture that contains a smaller proportion of the total multiplications within the input layer.

Acknowledgements

This work was supported by the Irish Research Council under their Government of Ireland Postgraduate Scholarship 2013.

References

[Courbariaux and Bengio, 2016] Courbariaux, M. and Bengio, Y. (2016). BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *arXiv*, page 9.

[Courbariaux et al., 2015] Courbariaux, M., Bengio, Y., and David, J.-P. (2015). BinaryConnect: Training Deep Neural Networks with binary weights during propagations. *Nips*, pages 1–9.

[Gong et al., 2014] Gong, Y., Liu, L., Yang, M., and Bourdev, L. (2014). Compressing Deep Convolutional Networks using Vector Quantization. pages 1–10.

[Hubara et al., 2016] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *Proceedings of the ACM International Conference on Multimedia - MM '14*, pages 675–678.

[Krizhevsky et al., 2012] Krizhevsky, A., Hinton, G. E., and Sutskever, I. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *the Neural Information Processing Systems Foundation 2012 conference*, pages 1097–1105.

[O'Keeffe and Villing, 2017] O'Keeffe, S. and Villing, R. (2017). A Benchmark Data Set and Evaluation of Deep Learning Architectures for Ball Detection in the RoboCup SPL. In *Accepted for publication at RoboCup 2017.*

[Rastegari et al., 2016] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *Eccv*, pages 1–17.

[Zhou et al., 2016] Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. (2016). DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv*, 1(1):1–14.