FIXED POINT QUANTIZATION OF DEEP CONVOLUTIONAL NETWORKS

Darryl D. Lin, Sachin S. Talathi
Qualcomm Research
{dexul, stalathi}@qti.qualcomm.com

V. Sreekanth Annapureddy
NetraDyne Inc.
sreekanthav@gmail.com

ABSTRACT

In recent years increasingly complex architectures for deep convolution networks (DCNs) have been proposed to boost the performance on image recognition tasks. However, the gains in performance have come at a cost of substantial increase in computation and model storage resources. Fixed point implementation of DCNs has the potential to alleviate some of these complexities and facilitate potential deployment on embedded hardware. In this paper, we formulate and solve an optimization problem to identify the optimal fixed point bit-width allocation across layers to enable efficient fixed point implementation of DCNs. Our experiments show that in comparison to equal bit-width settings, optimized bit-width allocation offers $> 20\%$ reduction in model size without any loss in accuracy on CIFAR-10 benchmark. We also demonstrate that fine-tuning can further enhance the accuracy of fixed point DCNs beyond that of the original floating point model. In doing so, we report a new state-of-the-art fixed point performance of 6.78% error-rate on CIFAR-10 benchmark.

1 INTRODUCTION AND RELATED WORK

Recent advances in the development of deep convolution networks (DCNs) have led to significant progress in solving non-trivial machine learning problems involving image recognition (Krizhevsky et al., 2012) and speech recognition (Deng et al., 2013). While increasing computational complexity has afforded improvements in the state-of-the-art performance, the added burden of training and testing times makes these networks impractical for real world applications involving real time processing and for deployment on mobile devices or embedded hardware with limited power budget. One of the approaches that may be cost efficient for large scale deployment is to implement DCNs in fixed point, which may offer advantages in reducing memory bandwidth, lowering power consumption and computation time as well as the storage requirements for the DCN models.

Earlier works in the area have primarily centered around designing the fixed point network during training (Courbariaux et al., 2014; Gupta et al., 2015; Lin et al., 2015). In contrast, we focus on developing a principled approach to converting a pre-trained floating point DCN model to its fixed point equivalent. This is important because in many real-world applications, a pre-trained DCN is used as a feature extractor, followed by a domain specific classifier or a regressor. In these applications, the user does not have access to the original training data and the training framework. For such cases, our proposed algorithm will offer an optimized method to convert any off-the-shelf pre-trained DCN model for efficient run time in fixed point. The works of Kyuyeon & Sung (2014); Sajid et al. (2015) more closely resemble our work. In the spirit of Sajid et al. (2015), we also focus on optimizing DCN models that are pre-trained with floating point precision. However, as opposed to exhaustive search method adopted by Sajid et al. (2015), our objective is to convert the pre-trained DCN model into a fixed point model using an optimization strategy based on signal-to-quantization noise ratio (SQNR). Other works that are somewhat closely related are Vanhoucke et al. (2011); Gong et al. (2014).

The proposed algorithm can be seen as a stand-alone tool to convert a floating point model to fixed point, and can be used in conjunction with other state-of-the-art network pruning and compression techniques such as Han et al. (2015).

2 FLOATING POINT TO FIXED POINT CONVERSION

There are three inter-dependent parameters to determine for the fixed point representation of a floating point DCN: bit-width, step-size (resolution), and dynamic range. They are related as:

$$\text{Range} \approx \text{Stepsize} \cdot 2^{\text{Bitwidth}} \tag{1}$$

Without fine-tuning, converting a floating point deep network to fixed point is essentially a process of introducing quantization noise into the network. It is well-understood in fields like audio processing or digital communications that the effect of quantization can be accurately captured in a single quantity, the SQNR. In particular, for an optimal uniform quantizer with ideal input, there is an approximately linear relationship between the bit-width and the resulting SQNR (You, 2010):

$$\gamma_{\text{dB}} \approx \kappa \cdot \beta \tag{2}$$

where, $\gamma_{\text{dB}} = 10\log_{10}(\gamma)$, is the SQNR in dB, κ is the quantization efficiency, and β is the bit-width. Suppose we try to perform uniform quantization of the weight value w, the quantized version $\tilde{w}$ can written as: $\tilde{w} = w + n_w$, where n_w is the quantization noise. According to Equation 2, the SQNR, γ_w, as a result of the quantization process can be written as:

$$10\log(\gamma_w) = 10\log\frac{\mathsf{E}[w^2]}{\mathsf{E}[n_w^2]} \approx \kappa \cdot \beta \tag{3}$$

Consider the case where both the weights and activations are quantized, it can be shown that the SQNR (γ_{output}) at the output of layer L of the network can be approximately expressed as:

$$\frac{1}{\gamma_{\text{output}}} = \frac{1}{\gamma_{a^{(0)}}} + \frac{1}{\gamma_{w^{(1)}}} + \frac{1}{\gamma_{a^{(1)}}} + \cdots + \frac{1}{\gamma_{w^{(L)}}} + \frac{1}{\gamma_{a^{(L)}}} \tag{4}$$

where γ_a and γ_w are the SQNR of activation and weight, respectively.

In other words, the SQNR at the output of a layer in DCN is the *Harmonic Mean* of the SQNRs of all preceeding quantization steps. This simple relationship reveals some very interesting observations: (1) All the quantization steps contribute equally to the overall SQNR of the output, regardless if it is the quantization of weights, activations, or input, and irrespective of its location in the network. (2) Since the output SQNR is the harmonic mean, the network performance will be dominated by the quantization step with the lowest precision. (3) Doubling the depth of a work will half the output SQNR (3dB loss). But this loss can be readily recovered by adding just 1 bit to the bit-width of all weights and activations, assuming the quantization efficiency is more than 3dB/bit.

3 CROSS-LAYER BIT-WIDTH OPTIMIZATION

From Equation 4, it is seen that trade-offs can be made between quantizers of different layers to produce the same γ_{output}. That is to say, we may choose to use smaller bit-widths for some layers by increasing bit-widths for other layers. This may be desirable because of the following reasons:

- Some layers may require a large number of computations (multiply-accumulate operations). Reducing the bit-widths for these layers would reduce the overall network computation load.
- Some layers may contain a large number of network parameters (weights). Reducing the weight bit-widths for these layers would reduce the overall model size.

Interestingly, such objectives can be formulated as an optimization problem and solved exactly. Suppose our goal is to reduce model size while maintaining a minimum SQNR at the DCN output. We can use ρ_i as the scaling factor at quantization step i. ρ_i represents the number of parameters being quantized in the quantization step. The problem can be written as:

$$\min \quad \textstyle\sum_i \rho_i \left(\frac{10\log\gamma_i}{\kappa}\right), \quad \text{s.t.} \quad \sum_i \frac{1}{\gamma_i} \leq \frac{1}{\gamma_{\min}} \tag{5}$$

where $10\log\gamma_i$ is the SQNR in dB domain, and $(10\log\gamma_i)/\kappa$ is the bit-width in the ith quantization step according to Equation 2. $\gamma_{\min}$ is the minimum output SQNR required to achieve a certain

level of accuracy. The optimization constraint follows from Equation 4 that the output SQNR is the harmonic mean of the SQNR of intermediate quantization steps. Substituting by $\lambda_i = \frac{1}{\gamma_i}$ and removing the constant scalars from the objective function, the problem can be reformulated as:

$$\min \quad -\sum_i \rho_i \log \lambda_i, \quad \text{s.t.} \quad \sum_i \lambda_i \leq C \tag{6}$$

where the constant $C = \frac{1}{\gamma_{\min}}$. This is a classic constrained optimization problem with a well-known solution: $\frac{\rho_i}{\lambda_i} = \text{constant}$. Or equivalently, $\rho_i \gamma_i = \text{constant}$. Recognizing that $10 \log \gamma_i = \kappa \beta_i$ based on Equation 2, the solution can be rewritten as:

$$\frac{10 \log \rho_i}{\kappa} + \beta_i = \text{constant} \tag{7}$$

In other words, the difference between the optimal bit-widths of two quantization steps is inversely proportional to the difference of ρ's in dB, scaled by the quantization efficiency.

$$\beta_i - \beta_j = \frac{10 \log(\rho_j / \rho_i)}{\kappa} \tag{8}$$

This is a surprisingly simple and intuitive relationship. For example, assuming $\kappa = 3$dB/bit, the bit-widths β_i and β_j would differ by 1 bit if ρ_j is 3dB (or 2x) larger than ρ_i. More specifically, for model size reduction, layers with more parameters should use relatively lower bit-width, as it leads to better model compression under the overall SQNR constraint.

4 Experiments

We evaluate our proposed cross-layer bit-width optimization algorithm on the CIFAR-10 benchmark. Consider the objective of minimizing the overall model size by applying Equation 8. Assuming $\kappa = 3$dB/bit, when the bit-width for layer conv0 is β_0, it can be shown that the optimal bit-widths of subsequent layers should be $\beta_0 - 5$ (conv1), $\beta_0 - 5$ (conv2), $\beta_0 - 6$ (conv3), $\beta_0 - 6$ (conv4), $\beta_0 - 8$ (conv5).

Figure 1 plots the model size vs. error rate in a scatter plot. We can clearly see the advantage of cross-layer bit-width optimization. When the model size is large (bit-width is high), the error rate saturates at around 6.9%. When the model size reduces below approximately 25Mbits, the error rate starts to increase quickly as the model size decreases. In this region, cross-layer bit-width optimization offers $> 20\%$ reduction in model size compared to equal bit-width allocation.

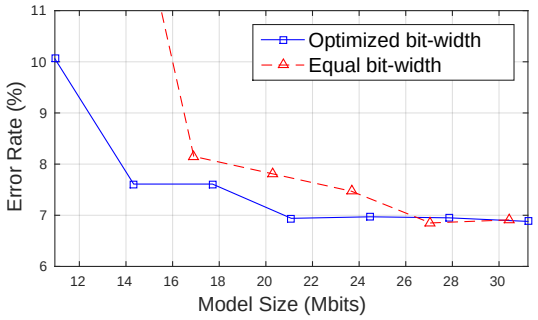

Figure 1: Model size vs. error rate with and without cross-layer bit-width optimization (CIFAR-10)

We conducted the same experiment on an AlexNet-like network for ImageNet classification. Similar to our CIFAR-10 network, there is a clear benefit of cross-layer bit-width optimization in terms of model size reduction. However, in the AlexNet-like DCN the convolutional layers comprises most of the complexity in terms of multiply-accumulate operations whereas the fully-connected layers dominate in the number of parameters. Therefore, the relative benefit of model size reduction is smaller in this case.

Although our focus is fixed point implementation after training, our quantizer design can also be used as a starting point for further model fine-tuning when the training model and data are available. When the network is fine-tuned after quantization, we achieve a new state-of-the-art CIFAR-10 classification error rate of 6.78% with floating point weights and 8-bit fixed point activations.

REFERENCES

M. Courbariaux, Y. Bengio, and J. David. Low precision arithmetic for deep learning. *arXiv:1412.7024*, 2014.

L. Deng, Hinton G.E., and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *IEEE International Conference on Acoustic, Speech and Signal Processing*, pp. 8599–8603, 2013.

Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv:1412.6115*, 2014.

S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. *arXiv:1502.02551*, 2015.

S. Han, H. Mao, and W. J. Dally. A deep neural network compression pipeline: Pruning, quantization, Huffman encoding. *arXiv:1510.00149*, 2015.

A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

H. Kyuyeon and W. Sung. Fixed-point feedforward deep neural network design using weights +1, 0 and -1. In *IEEE Workshop on Signal Processing Systems (SiPS)*, 2014.

Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio. Neural networks with few multiplications. *arXiv:1510.03009*, 2015.

A. Sajid, H. Kyuyeon, and W. Sung. Fixed point optimization of deep convolutional neural networks for object recognition. In *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2015.

V. Vanhoucke, A. Senior, and M. Mao. Improving the speed of neural networks on CPUs. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, 2011.

Yuli You. *Audio Coding: Theory and Applications*. Springer, 2010.