FIXED POINT QUANTIZATION OF DEEP CONVOLUTIONAL NETWORKS

Darryl D. Lin, Sachin S. Talathi & V. Sreekanth Annapureddy
Qualcomm Research
San Diego, CA 92121, USA
{dexul,stalathi,venkataa}@qti.qualcomm.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 compute resources, the model size and processing speed of the network for training and evaluation. Fixed point implementation of these networks has the potential to alleviate some of the burden of these additional complexities. In this paper, we propose a quantizer design for fixed point implementation for DCNs. We then formulate an optimization problem to identify optimal fixed point bit-width allocation across DCN layers. We perform experiments on a recently proposed DCN architecture for CIFAR-10 benchmark that generates test error of less than 7%. We evaluate the effectiveness of our proposed fixed point bit-width allocation for this DCN. Our experiments show that in comparison to equal bit-width settings, the fixed point DCNs with optimized bit width allocation offer $> 20\%$ reduction in the model size without any loss in performance. 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

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). Over the last two years several advances in the design of DCNs (Zeiler & Fergus, 2014; Simonyan & Zisserman, 2014; Szegedy et al., 2014; Chatfield et al., 2014; He et al., 2014; Ioffe & Szegedy, 2015) have not only led to a further boost in achieved accuracy on image recognition tasks but also have played crucial role as a feature generator for other machine learning tasks such as object detection (Krizhevsky et al., 2012) and localization (Sermanet et al., 2013), semantic segmentation (Girshick et al., 2014) and image retrieval (Krizhevsky et al., 2012; Razavian et al., 2014). These advances have come with an added cost of computational complexity, resulting from DCN designs involving any combinations of: increasing the number of layers in the DCN (Szegedy et al., 2014; Simonyan & Zisserman, 2014; Chatfield et al., 2014), increasing the number of filters per convolution layer (Zeiler & Fergus, 2014), decreasing stride per convolution layer (Sermanet et al., 2013; Simonyan & Zisserman, 2014) and hybrid architectures that combine various DCN layers (Szegedy et al., 2014; He et al., 2014; Ioffe & Szegedy, 2015).

While increasing computational complexity has afforded improvements in the state-of-the-art performance, the added burden of training and testing times make these networks impractical for real world applications that involve real time processing and for deployment on mobile devices with limited power budget. One approach to alleviate this burden is to increase the computational power of the hardware used to deploy these networks. An alternative approach 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 DCNs.

This paper investigates methods to achieve fixed point implementation of DCN. The paper is organized as follows: In Section 2, we present a literature survey of the related work. In Section 3, we develop quantizer design for fixed point DCNs. In Section 4 we formulate an optimization problem to identify optimal fixed point bit-width allocation per layer of DCNs to maximize the achieved reduction in complexity relative to the loss in the classification accuracy of the DCN model. Results from our experiments with one of the state-of-the-art floating point DCNs for CIFAR-10 benchmark are reported in Section 5 followed by conclusions in the last section.

2 RELATED WORK

Fixed width implementation of DCN have been explored in earlier works (Courbariaux et al., 2014; Gupta et al., 2015). These works primarily focused on training DCNs using low precision fixed-point arithmetic. More recently, (Lin et al., 2015) showed that deep neural networks can be effectively trained using only binary weights, which in some cases can even improve classification accuracy relative to the floating point baseline.

Here, we focus on optimizing DCN models that are pre-trained with floating point precision. Our objective is to convert the pre-trained DCN model into a fixed-point model to improve upon the inference speed of the network and reduce storage requirements. There has been some work in literature related to our stated objective. For example, Vanhoucke et al. (2011) quantized the weights and activations of pre-trained deep networks using 8-bit fixed-point representation to improve inference speed. Han et al. (2015) and Gong et al. (2014) on the other hand applied codebook based on scalar and vector quantization methods respectively, in order to reduce the model size. Our work is different from these works in the following sense:

- We use different bitwidths across layers, and propose a method for bitwidth allocation.
- We use signal-to-quantization noise (SQNR) ratio as a proxy for classification performance.
- We use quantizer step sizes derived from optimal quantizer design theory.

3 FLOATING POINT TO FIXED POINT CONVERSION

In this section, we will propose an algorithm to convert a floating point DCN to fixed point. For a given layer of DCN the goal of conversion is to represent the input activations, the output activations, and the parameters of that layer in fixed point. This can be seen as a process of quantization.

3.1 OPTIMAL UNIFORM QUANTIZER

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. These are related as follows:

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

Given a fixed bit-width, the trade-off is between having large enough range to reduce the chance of overflow and small enough resolution to reduce the quantization error. The problem of striking the best trade-off between overflow error and quantization error has been extensively studied in the literature. Table 1 below shows the step sizes of the optimal symmetric uniform quantizer for Gaussian, Laplacian and Gamma distributions. The quantizers are optimal in the sense of minimizing the signal-to-quantization-noise ratio (SQNR).

For example, suppose the input is Gaussian distributed with zero mean and unit variance. If we need a uniform quantizer with bit-width of 1 (i.e. 2 levels), the best approach is to place the quantized values at -0.798 and 0.798. In other words, the step size is 1.596. If we need a quantizer with bit-width of 2 (i.e. 4 levels), the best approach is to place the quantized values at -1.494, -0.498, 0.498, and 1.494. In other words, the step size is 0.996.

In practice, however, even though a symmetric quantizer is optimal for a symmetric input distribution, it is often desirable to have 0 as one of the quantized values because of the potential savings in model storage and computational complexity. This means that for a quantizer with 2 levels, the

Table 1: Step-sizes of optimal symmetric uniform quantizer for various input distributions (Shi & Sun, 2008)

Bit-width β	**Uniform**	**Gaussian**	**Laplacian**	**Gamma**
1	1.0	1.596	1.414	1.154
2	0.5	0.996	1.087	1.060
3	0.25	0.586	0.731	0.796
4	0.125	0.335	0.456	0.540

quantized values would be 0 and 1.596 (or -1.596 and 0). While this is clearly suboptimal when the number of quantized levels is small, it performs increasingly closer to a symmetric quantizer when the bit-width, $\beta >> 1$.

Given an optimal uniform quantizer with ideal input, the resulting SQNR as a function of the bit-width is shown in Figure 1. It can be observed that the quantization efficiency decreases as the Kurtosis of the input distribution increases.

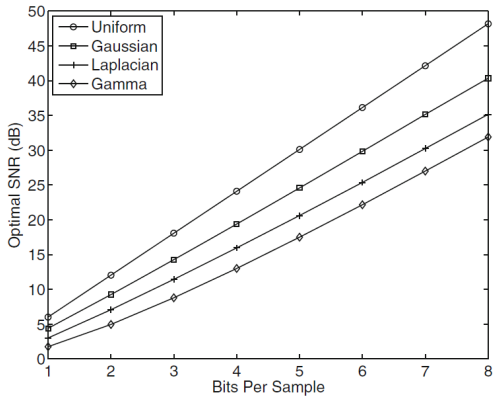


Figure 1: Optimal SQNR achieved by uniform quantizer for uniform, Gaussian, Laplacian and Gamma distributions (You, 2010)

Another take-away from this figure is that there is an approximately linear relationship between the bit-width and resulting SQNR:

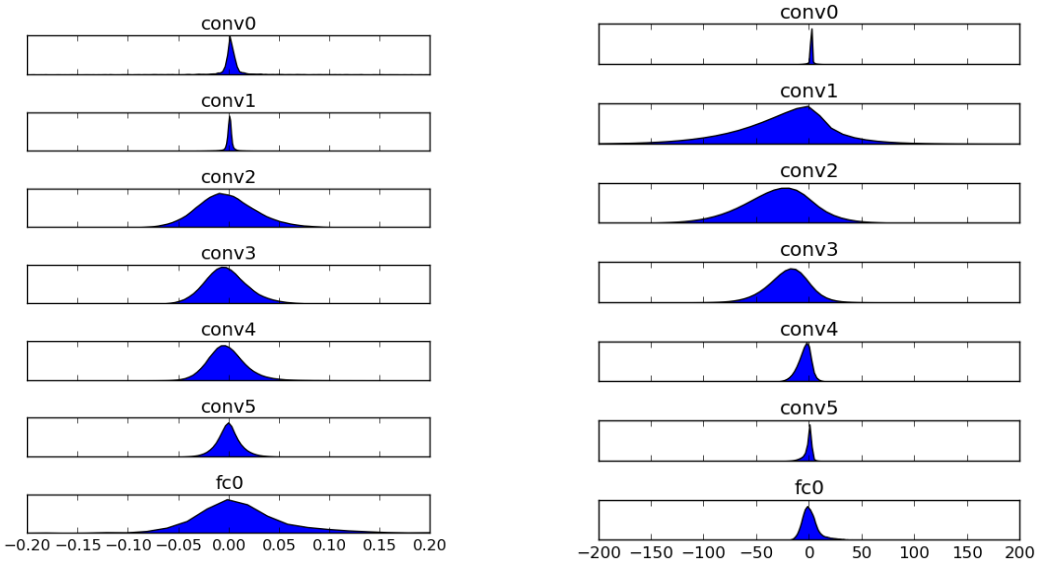
$$\gamma_{\text{dB}} \approx \kappa \cdot \beta \tag{2}$$

where, $\gamma_{\text{dB}} = 10 \log_{10}(\gamma)$, is the SQNR in dB, κ is the quantization efficiency, and β is the bit-width. Note that the slopes of the lines in Figure 1 depict the optimal quantization efficiency for ideal distributions. The quantization efficiency for uniform distribution is the well-known value of 6dB/bit (Shi & Sun, 2008), while the quantization efficiency for Gaussian distribution is about 5dB/bit (You, 2010) . Actual quantization efficiency for non-ideal inputs can be significantly lower. Our experiments show that the SQNR resulting from uniform quantization of the actual weights and activations in the DCN is between 2 to 4dB/bit.

3.2 Empirical distributions in DCN

Our simulations have revealed some interesting statistics for DCNs. Figure 2a and Figure 2b depict the empirical distributions of weights and activations, respectively for the CIFAR-10 benchmark DCN (see Section 5).

Note that the activations plotted here are before applying the activation functions. Taking the ReLU neurons for example, the activations at the output of the neuron would not have any negative values. It is seen that in most cases, both the weights and activations are roughly Gaussian distributed. The activations of earlier layers have longer tails, making them resemble the Laplacian distributions. In



(a) Histogram of weight values

(b) Histogram of output activation values

Figure 2: Distribution of weights & activations in a DCN design for CIFAR-10 benchmark.

our fixed point design, Gaussian distribution is typically assumed. Since the weight and activation distributions of some layers are less Gaussian-like, it is also of interest to experiment with step-sizes for other distributions (see Table 1), which is beyond the scope of present work.

3.3 Model conversion

Any floating point DCN model can be converted to fixed point by following these steps:

- Run a forward pass in floating point using a sufficiently large set of typical inputs and record the activations
- Collect the statistics of weights, biases and activations for each layer
- Determine the number formats of the weights, biases and activations for each layer

Note that determining the number format for a fixed point quantity is equivalent to determining the resolution, which in turn means identifying the number of fractional bits it requires to represent the number. The following equations can be used to compute the number of fractional bits:

- Determine the effective standard deviation of the quantity being quantized: $\hat{\sigma}$
- Calculate step size according to Table 1: $s = \hat{\sigma} \cdot \text{Stepsize}(\beta)$
- Compute the number of fractional bits: $n = -\lceil \log_2 s \rceil$

In these equations,

- $\hat{\sigma}$ is the effective standard deviation of the quantity being quantized, an indication of the width of the distribution we want to quantize. For example, if the quantized quantities follow an ideal zero mean Gaussian distribution, then $\hat{\sigma} = \sigma$, where σ is the true standard deviation of quantized values. If the actual distribution has longer tails than Gaussian, which is often the case as shown in Figure 2a and 2b, then $\hat{\sigma} > \sigma$. In our experiments in Section 5, we set $\hat{\sigma} = 3\sigma$.
- $\text{Stepsize}(\beta)$ is the optimal step size corresponding to quantization bit-width of β, as listed in Table 1.
- s is the computed step size for the quantized distribution.

- n is the number of fractional bits in the fixed point representation. Equivalently, 2^{-n} is the resolution of the fixed point representation and a quantized version of s. Note that $\lceil \cdot \rfloor$ is one choice of a rounding function and is not unique.

4 BIT-WIDTH OPTIMIZATION ACROSS A DEEP NETWORK

In the absence of model fine-tuning, converting a floating point deep network into a fixed point deep network is essentially a processing of introducing quantization noise into the neural network. It is well understood in fields like audio processing or digital communications that as the quantization noise increases, the system performance degrades. The effect of quantization can be accurately captured in a single quantity, the SQNR.

In deep learning, there is not a well-formulated relationship between SQNR and classification accuracy, but it is reasonable to assume that in general higher quantization noise level leads to worse classification performance. Therefore, as shown in Figure 3, to simplify our study we will make an approximating conjecture that the SQNR of the output activations used for classification is a surrogate measure for the final accuracy.

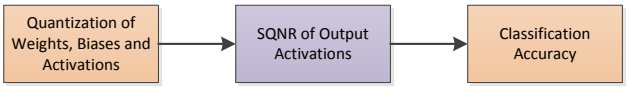


Figure 3: SQNR as a measure of how quantization affects accuracy

While the relationship between SQNR and classification accuracy is not exactly monotonic, as our experiments in Section 5.1 indicate, it is a close approximation. This approximation is also important because it is usually difficult to directly predict the classification accuracy. SQNR, on the other hand, can be approximated theoretically and analyzed layer by layer, as can be seen in the next section.

4.1 IMPACT OF QUANTIZATION ON SQNR

In this section, we will derive the relationship between the quantization of the weight, bias and activation values respectively, and the resulting output SQNR.

4.1.1 QUANTIZATION OF INDIVIDUAL VALUES

Quantization of individual values in a DCN, whether it is an activation or weight value, readily follows the quantizer discussion in Section 3.1. For instance, for weight value w, the quantized version $\tilde{w}$ can written as:

$$\tilde{w} = w + n_w, \tag{3}$$

where n_w is the quantization noise. As illustrated in Figure 1, if w is approximately follows a Gaussian, Laplacian or Gamma distribution, the SQNR, γ_w, as a result of the quantization process can be written as:

$$\gamma_w = 10 \log \frac{\mathsf{E}[w^2]}{\mathsf{E}[n_w^2]} \cong \kappa \cdot \beta, \tag{4}$$

where κ is the quantization efficiency and β is the quantizer bit-width.

4.1.2 QUANTIZATION OF BOTH ACTIVATIONS AND WEIGHTS

Consider the case where weight w is multiplied by activation a, where both w and a are quantized with quantization noise n_w and n_a, respectively. The product can be approximated, for small n_w and n_a, as follows:

$$\begin{aligned} \tilde{w} \cdot \tilde{a} &= (w + n_w) \cdot (a + n_a) \\ &= w \cdot a + w \cdot n_a + n_w \cdot a + n_w \cdot n_a \\ &\cong w \cdot a + w \cdot n_a + n_w \cdot a. \end{aligned} \tag{5}$$

The last equality holds if $|n_a| << |a|$ and $|n_w| << |w|$. A very important observation is that the SQNR of the product, $w \cdot a$, as a result of quantization, satisfies

$$\frac{1}{\gamma_{w \cdot a}} = \frac{1}{\gamma_w} + \frac{1}{\gamma_a}. \tag{6}$$

This is characteristic of a linear system. The defining benefit of this realization is that introducing quantization noise to weights and activations independently is equivalent to adding the total noise after the product operation in a normalized system. This property will be used in later analysis.

4.1.3 FORWARD PASS THROUGH ONE LAYER

In a DCN with multiple layers, computation of the ith activation in layer $l+1$ of the DCN can be expressed as follows:

$$a_i^{(l+1)} = \sum_{j=1}^{N} w_{i,j}^{(l+1)} a_j^{(l)} + b_i^{(l+1)}, \tag{7}$$

where (l) represents the lth layer, N represents number of additions, $w_{i,j}$ represents the weight and b_i represents the bias.

Ignoring the bias term for the time being, since $a_i^{(i+1)}$ is simply a sum of terms like $w_{i,j}^{(l+1)} a_j^{(l)}$, which when quantized all have the same SQNR $\gamma_{w^{(l+1)} \cdot a^{(l)}}$. Assuming the product terms $w_{i,j}^{(l+1)} a_j^{(l)}$ are independent, it follows that the value of $a_i^{(i+1)}$, before further quantization, has inverse SQNR that equals

$$\frac{1}{\gamma_{w_{i,j}^{(l+1)} a_j^{(l)}}} = \frac{1}{\gamma_{w_{i,j}^{(l+1)}}} + \frac{1}{\gamma_{a_j^{(l)}}} = \frac{1}{\gamma_{w^{(l+1)}}} + \frac{1}{\gamma_{a^{(l)}}} \tag{8}$$

After $a_i^{(l+1)}$ is quantized to the assigned bit-width, the resulting inverse SQNR then becomes $\frac{1}{\gamma_{a^{(l+1)}}} + \frac{1}{\gamma_{w^{(l+1)}}} + \frac{1}{\gamma_{a^{(l)}}}$. We are not considering the biases in this analysis because, assuming that the biases are quantized at the same bit-width as the weights, the SQNR is dominated by the product term $w_{i,j}^{(l+1)} a_j^{(l)}$. Note that Equation 8 matches rather well with experiments even though the independence assumption of $w_{i,j}^{(l+1)} a_j^{(l)}$ does not always hold.

4.1.4 FORWARD PASS THROUGH THE ENTIRE NETWORK

Equation 8 can be generalized to all the convolutional and fully-connected layers in a DCN. Consider a deep network with L such layers as shown in Figure 5 in Appendix 7.1, where all the activations and weights are quantized. Extending Equation 8, we obtain the SQNR (γ_{output}) at the output of the DCN and input to the classifier as:

$$\frac{1}{\gamma_{\text{output}}} = \frac{1}{\gamma_{a^{(0)}}} + \frac{1}{\gamma_{w^{(1)}}} + \frac{1}{\gamma_{a^{(1)}}} + \frac{1}{\gamma_{w^{(2)}}} + \cdots + \frac{1}{\gamma_{w^{(L)}}} + \frac{1}{\gamma_{a^{(L)}}} \tag{9}$$

In other word, the SQNR at the output of the DCN is the *Harmonic Mean* of the SQNRs of all quantization steps. This simple relationship reveals some very interesting insights:

- All the quantization steps contribute equally to the overall SQNR of the output, regardless if it's the quantization of weights, activations, or input, and irrespective of where it happens (at the top or bottom of the network).
- Since the output SQNR is the harmonic mean. The network performance will be dominated by the worst quantization step. For example, if the activations of a particular layer has a much smaller bit-width than other layers, it will be bottleneck of network performance, because based on Equation 9, $\gamma_{\text{output}} \leq \gamma_{a^{(l)}}$ for all l.
- Depth makes quantization more challenging, but not exceedingly so. The rest being the same, doubling the depth of a DCN will half the output SQNR (3dB loss). But this loss can be readily recovered by adding 1 bit to the bit-width of all weights and activations, assuming the quantization efficiency is more than 3dB/bit. However, this theoretical prediction will need to be empirically verified in future works.

4.2 CROSS-LAYER BIT-WIDTH OPTIMIZATION

From Equation 9, 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 can choose to use smaller bit-widths for some layers by increasing bit-widths for other layers. For example, this may be desirable because of the following reasons:

- Some layers may require large number of computations. Reducing the bit-widths for these layers would reduce the overall network computation load.
- Some layers may contain 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, which in this case represents the number of parameters being quantized in the quantization step. The problem can be written as:

$$\min \quad \sum_i \rho_i \left(\frac{10 \log \gamma_i}{\kappa} \right) \tag{10}$$

$$\text{s.t.} \quad \sum_i \frac{1}{\gamma_i} \leq \frac{1}{\gamma_{\min}} \tag{11}$$

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 summation of γ_i's follows from Equation 9 that the output SQNR is the harmonic mean of the SQNR of intermediate quantization steps.

Substituting by $\lambda_i = \dfrac{1}{\gamma_i}$ and removing the constant scalars from the objective function, the problem can be reformulated as:

$$\min \quad -\sum_i \rho_i \log \lambda_i \tag{12}$$

$$\text{s.t.} \quad \sum_i \lambda_i \leq C \tag{13}$$

where the constant $C = \dfrac{1}{\gamma_{\min}}$. This is a classic constrained optimization problem with a well-known solution: $\dfrac{\rho_i}{\lambda_i} = \text{constant}$. Or equivalently,

$$\rho_i \gamma_i = \text{constant} \tag{14}$$

Recognizing that $10 \log \gamma_i = \kappa \beta_i$ based on Equation 2, the solution can be rewritten as:

$$\frac{10 \log \rho_i}{\kappa} + \beta_i = \text{constant} \tag{15}$$

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 quantization efficiency.

$$\beta_i - \beta_j = \frac{10 \log(\rho_j / \rho_i)}{\kappa} \tag{16}$$

This is a surprisingly simple and insightful 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. In other words, for model size reduction, layers with more parameters should use relatively lower bit-width, because it leads to better model compression under the overall SQNR constraint.

5 EXPERIMENTS

In this section we study the effect of reduced bit-width for both weights and activations versus traditional 32-bit single-precision floating point approach. In particular, we will implement the fixed point quantization algorithm described in Section 3 and investigate the effectiveness of the bit-width optimization algorithm in Section 4. In addition, using the quantized fixed point network as the starting point, we will attempt to further fine-tune the fixed point network within the restricted alphabets of weight and activation values.

5.1 FIXED POINT QUANTIZATION ALGORITHM

We evaluate our proposed floating-point to fixed-point conversion algorithm on the CIFAR-10 benchmark. The description of the DCN we have designed can be found in Appendix 7.2.

Table 2 lists the classification error rate (in %) for both floating point and 4-bit, 8-bit, and 16-bit fixed point weight and activation bit-width combinations for the CIFAR-10 network.

Table 2: CIFAR-10 performance with different bit-width combinations

	Weight Bit-width							
	No Fine-tuning				With Fine-tuning			
Activation Bit-width	4	8	16	Float	4	8	16	Float
4	27.68	9.90	9.69	9.73	8.30	7.50	7.40	7.44
8	18.15	7.55	7.49	7.47	7.58	6.95	6.95	6.78
16	15.04	6.98	6.92	6.91	7.58	6.82	6.92	6.83
Float	14.90	7.02	6.98	6.98	7.62	6.94	6.96	6.98

It is seen that, using the proposed quantization algorithm with no fine-tuning, the quantized classifiers work well down to (8b, 8b) combination (8-bit weights and 8-bit activations) and generally degrade as bit-width decreases, but (16b, 16b) and (float, 16b) fixed point models are even better than the floating point baseline. We will discuss the results with fine-tuning in Section 5.3.

5.2 BIT-WIDTH OPTIMIZATION ACROSS LAYERS

In Section 5.1, we focused on scenarios where the same bit-width is shared by weights (or activations) across all layers. In this section, we will look into how to further optimize the classification performance via the cross-layer bit-width optimization algorithm prescribed in Section 4.2.

Table 3: Number of parameters per layer in our CIFAR-10 network

Layer	Input channels	Output channels	Output maps	Filter dimension	Parameters ($\times 10^6$)
conv0	3	256	22	3	0.007
conv1	256	128	12	3	0.295
conv2	128	256	10	3	0.295
conv3	256	256	8	3	0.590
conv4	256	256	5	3	0.590
conv5	256	128	2	7	1.606
fc0	128	10	1	1	0.005

In Table 3, we compute the number of parameters in each layer of our CIFAR-10 network. Consider the objective of minimizing the overall model size. Our derivation in Section 4.2 shows that, provided that the quantization efficiency $\kappa = 3$dB/bit, the optimal bit-width of layer conv0 and conv1 would differ by $10\log(0.295/0.007)/\kappa = 5$bits. Similarly, assuming the bit-width for layer conv0 is β_0, the subsequent convolutional layers should have bit-width values as indicated in Table 4.

Table 4: Optimal bit-width allocation in our CIFAR-10 network

Layer	conv0	conv1	conv2	conv3	conv4	conv5
Bit-width	β_0	$\beta_0 - 5$	$\beta_0 - 5$	$\beta_0 - 6$	$\beta_0 - 6$	$\beta_0 - 8$

In our experiment in this section, we will ignore the fully-connected layer and assume a fixed bit-width of 16 for both weights and activations. This is because fully-connected layers have different SQNR characteristics and need to be optimized separately. Although the fully connected layers can often be quantized more aggressively than convolutional layers, in this experiment, since the number of parameters of fc0 is very small, we will set the bit-width to a large value to eliminate the impact of quantizing fc0 from the analysis, knowing that the large bit-width of fc0 has very little impact on the overall model size. We will also set the activation bit-widths of all the layers to a large value of 16 because they contribute little to the model size.

Table 6 in Appendix 7.3 lists a number of bit-width allocation schemes where the weights of different layers either share the same bit-width or are optimized according to Table 4. The last column shows the classification error rate for each scenario.

Figure 4 displays 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, across-layer bit-width optimization offers $> 20\%$ reduction in model size for the same performance.

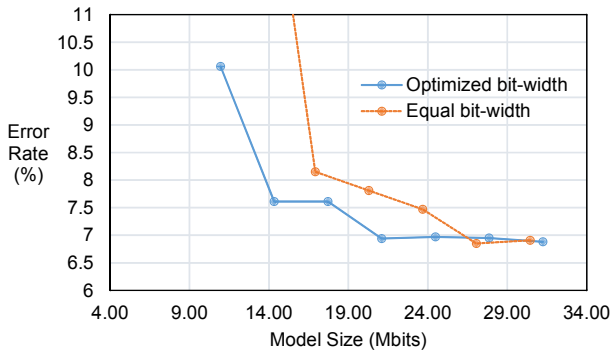


Figure 4: Model size vs. error rate with and without cross-layer bit-width optimization

5.3 Model fine-tuning

Table 2 also contains the classification error rate (in %) for the CIFAR-10 network after fine-tuning the model for 30 epochs. It is shown that even the (4b, 4b) bit-width combination works well (8.30% error rate) when the network is fine-tuned after quantization. In addition, the (float, 8b) setting generates an error rate of 6.78%, which is the new state-of-the-art result even though the activations are only 8-bit fixed point values. This may be attributed to the regularization effect of added noise, which reduces overfitting (Lin et al., 2015; Luo & Yang, 2014).

6 Conclusions and Future Work

We showed that quantizing all the layers in a neural network with the same uniform bit-width value is not the optimal strategy. We formulated an SQNR based optimization problem with a closed-form solution for bit-width allocation across layers. We showed that the proposed method achieves lower classification error on CIFAR10 dataset compared to the strategy of allocating uniform bit-width for all layers. We also performed fine tuning experiments and demonstrate that even with a non-optimal bit-width allocation strategy, classification accuracy of DCN can be boosted with fine-tuning. It will be interesting to determine the effect of fine-tuning on a fixed-point implementation of DCN that follows the optimal strategy. Furthermore, It will be interesting to study the relation between SQNR and classification performance in a closer detail. Specifically, the relation between the depth of network and quantization is interesting. The SQNR formulation suggests using an additional bit on doubling the depth of the neural network. We plan to investigate these questions in future works.

ACKNOWLEDGEMENTS

We would like to acknowledge fruitful discussions and valuable feedback from our colleagues at Qualcomm Research: David Julian, Anthony Sarah, Daniel Fontijne, Somdeb Majumdar, Aniket Vartak, Blythe Towal, and Mark Staskauskas.

REFERENCES

Chatfield, K., Simonyan, S., Vedaldi, V., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv:1405:3531*, 2014.

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

Deng, L., G.E., Hinton, and Kingsbury, B. 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.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

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

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

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

He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv:1406:4729*, 2014.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502:03167*, 2015.

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

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

Luo, Y. and Yang, F. Deep learning with noise. *http://www.andrew.cmu.edu/user/fanyang1/DLWN.pdf*, 2014.

Razavian, A. S., Azizpour, H., Sullivan, J., and S., Carlsson. Cnn features off-the-shelf: An astounding baseline for recogniton. In *CVPR*, 2014.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312:6229*, 2013.

Shi, Yun Q. and Sun, Huifang. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*. CRC Press, 2008.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv:1409:1556*, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *arXiv:1409.4842*, 2014.

Talathi, S.S. Hyper-parameter optimization of deep convolutional networks for object recognition. *arXiv:1501.07645*, 2015.

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

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

Zeiler, M.D. and Fergus, R. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.

7 APPENDIX

7.1 DESCRIPTION OF LAYERS IN A DCN

Figure 5 provides a graphical illustration of the layered structure of a typical DCN.

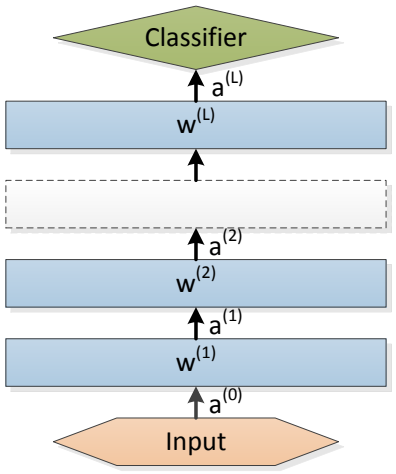


Figure 5: Layers in a DCN

7.2 DESCRIPTION OF OUR DCN FOR CIFAR-10 DATASET

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Of the 60,000 images, 50,000 are training images and 10,000 are test images.

Our floating point model produces the state-of-the-art 6.98% error rate with pReLU non-linearity (Talathi, 2015) after training of 550 epochs. The DCN architecture is summarized in Table 5.

Table 5: DCN architecture for CIFAR-10 dataset

	Convolutional						**Fully-connected**
Parameters	conv0	conv1	conv2	conv3	conv4	conv5	fc0
Number of Filters	256	128	256	256	256	128	10
Kernel Size	3x3	3x3	3x3	3x3	3x3	7x7	
Stride	1	2	1	1	2	5	
Padding	0	1	0	0	1	2	
Normalization	✓					✓	

7.3 BIT-WIDTH ALLOCATION SCENARIOS FOR CROSS-LAYER OPTIMIZATION

Table 6 lists a number of bit-width allocation schemes where the weights of different layers either share the same bit-width or are optimized according to Table 4. The last column shows the classification error rate for each scenario.

Table 6: Bit-width allocation scenarios and corresponding classification accuracy

	conv0	conv1	conv2	conv3	conv4	conv5	Model size ($\times 10^6$ bits)	Error rate (%)
Case #1	9	9	9	9	9	9	30.44	6.91
Case #2	8	8	8	8	8	8	27.06	6.85
Case #3	7	7	7	7	7	7	23.67	7.47
Case #4	6	6	6	6	6	6	20.29	7.81
Case #5	5	5	5	5	5	5	16.91	8.15
Case #6	4	4	4	4	4	4	13.53	15.04
Case #7	16	11	11	10	10	8	31.24	6.88
Case #8	15	10	10	9	9	7	27.86	6.95
Case #9	14	9	9	8	8	6	24.48	6.97
Case #10	13	8	8	7	7	5	21.09	6.94
Case #11	12	7	7	6	6	4	17.71	7.61
Case #12	11	6	6	5	5	3	14.33	7.61
Case #13	10	5	5	4	4	2	10.95	10.06