Adaptive Quantization for Deep Neural Network

Yiren Zhou[1], Seyed-Mohsen Moosavi-Dezfooli[2], Ngai-Man Cheung[1], Pascal Frossard[2]
[1]Singapore University of Technology and Design (SUTD)
[2]École Polytechnique Fédérale de Lausanne (EPFL)
yiren_zhou@mymail.sutd.edu.sg, ngaiman_cheung@sutd.edu.sg
{seyed.moosavi, pascal.frossard}@epfl.ch

Abstract

In recent years Deep Neural Networks (DNNs) have been rapidly developed in various applications, together with increasingly complex architectures. The performance gain of these DNNs generally comes with high computational costs and large memory consumption, which may not be affordable for mobile platforms. Deep model quantization can be used for reducing the computation and memory costs of DNNs, and deploying complex DNNs on mobile equipment. In this work, we propose an optimization framework for deep model quantization. First, we propose a measurement to estimate the effect of parameter quantization errors in individual layers on the overall model prediction accuracy. Then, we propose an optimization process based on this measurement for finding optimal quantization bit-width for each layer. This is the first work that theoretically analyse the relationship between parameter quantization errors of individual layers and model accuracy. Our new quantization algorithm outperforms previous quantization optimization methods, and achieves 20-40% higher compression rate compared to equal bit-width quantization at the same model prediction accuracy.

Introduction

Deep neural networks (DNNs) have achieved significant success in various machine learning applications, including image classification (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; Szegedy et al. 2015), image retrieval (Hoang et al. 2017; Do, Doan, and Cheung 2016), and natural language processing (Deng, Hinton, and Kingsbury 2013). These achievements come with increasing computational and memory cost, as the neural networks are becoming deeper (He et al. 2016), and contain more filters per single layer (Zeiler and Fergus 2014).

While the DNNs are powerful for various tasks, the increasing computational and memory costs make it difficult to apply on mobile platforms, considering the limited storage space, computation power, energy supply of mobile devices (Han, Mao, and Dally 2015), and the real-time processing requirements of mobile applications. There is clearly a need to reduce the computational resource requirements of DNN models so that they can be deployed on mobile devices (Zhou et al. 2016).

In order to reduce the resource requirement of DNN models, one approach relies on model pruning. By pruning some parameters in the model (Han, Mao, and Dally 2015), or skipping some operations during the evaluation (Figurnov et al. 2016), the storage space and/or the computational cost of DNN models can be reduced. Another approach consists in parameter quantization (Han, Mao, and Dally 2015). By applying quantization on model parameters, these parameters can be stored and computed under lower bit-width. The model size can be reduced, and the computation becomes more efficient under hardware support (Han et al. 2016). It is worth noting that model pruning and parameter quantization can be applied at the same time, without interfering with each other (Han, Mao, and Dally 2015); we can apply both approaches to achieve higher compression rates.

Many deep model compression works have also considered using parameter quantization (Gupta et al. 2015; Han, Mao, and Dally 2015; Wu et al. 2016) together with other compression techniques, and achieve good results. However, these works usually assign the same bit-width for quantization in the different layers of the deep network. In DNN models, the layers have different structures, which lead to the different properties related to quantization. By applying the same quantization bit-width for all layers, the results could be sub-optimal. It is however possible to assign different bit-width for different layers to achieve optimal quantization result (Hwang and Sung 2014).

In this work, we propose an accurate and efficient method to find the optimal bit-width for coefficient quantization on each DNN layer. Inspired by the analysis in (Fawzi, Moosavi-Dezfooli, and Frossard 2016), we propose a method to measure the effect of parameter quantization errors in individual layers on the overall model prediction accuracy. Then, by combining the effect caused by all layers, the optimal bit-width is decided for each layer. By this method we avoid the exhaustive search for optimal bit-width on each layer, and make the quantization process more efficient. We apply this method to quantize different models that have been pre-trained on ImageNet dataset and achieve good quantization results on all models. Our method constantly outperforms recent state-of-the-art, i.e., the SQNR-based method (Lin, Talathi, and Annapureddy 2016) on different models, and achieves 20-40% higher compression rate compared to equal bit-width quantization. Furthermore, we give a theoretical analysis on how the quantization on layers affects DNN accuracy. To the best of our knowledge, this is the first work that theoretically analyses the relationship between coefficient quantization effect of individual layers and DNN accuracy.

Related works

Parameter quantization has been widely used for DNN model compression (Gupta et al. 2015; Han, Mao, and Dally 2015; Wu et al. 2016). The work in (Gupta et al. 2015) limits the bit-width of DNN models for both training and testing, and stochastic rounding scheme is proposed for quantization to improve the model training performance under low bit-width. The authors in (Han, Mao, and Dally 2015) use k-means to train the quantization centroids, and use these centroids to quantize the parameters. The authors in (Wu et al. 2016) separate the parameter vectors into sub-vectors, and find sub-codebook of each sub-vectors for quantization. In these works, all (or a majority of) layers are quantized with the same bit-width. However, as the layers in DNN have various structures, these layers may have different properties with respect to quantization. It is possible to achieve better compression result by optimizing quantization bit-width for each layer.

Previous works have been done for optimizing quantization bit-width for DNN models (Hwang and Sung 2014; Anwar, Hwang, and Sung 2015; Lin, Talathi, and Annapureddy 2016; Sun, Lin, and Wang 2016). The authors in (Hwang and Sung 2014) propose an exhaustive search approach to find optimal bit-width for a fully-connected network. In (Sun, Lin, and Wang 2016), the authors first use exhaustive search to find optimal bit-width for uniform or non-uniform quantization; then two schemes are proposed to reduce the memory consumption during model testing. The exhaustive search approach only works for a relatively small network with few layers, while it is not practical for deep networks. As the number of layers increases, the complexity of exhaustive search increases exponentially. The authors in (Anwar, Hwang, and Sung 2015) use mean square quantization error (MSQE) (L_2 error) on layer weights to measure the sensitivity of DNN layers to quantization, and manually set the quantization bit-width for each layer. The work in (Lin, Talathi, and Annapureddy 2016) use the signal-to-quantization-noise ratio (SQNR) on layer weights to measure the effect of quantization error in each layer. These MSQE and SQNR are good metrics for measuring the quantization loss on model weights. However, there is no theoretical analysis to show how these measurements relate to the accuracy of the DNN model, but only empirical results are shown. The MSQE-based approach in (Anwar, Hwang, and Sung 2015) minimizes the L_2 error on quantized weight, indicating that the L_2 error in different layer has the equal effect on the model accuracy. Similarly, in (Lin, Talathi, and Annapureddy 2016), the authors maximize the overall SQNR, and suggest that quantization on different layers has equal contribution to the overall SQNR, thus has equal effect on model accuracy. Both works ignore that the various structure and position of different layers may lead to different robustness on quantization, and thus render the two approaches suboptimal.

In this work, we follow the analysis in (Fawzi, Moosavi-Dezfooli, and Frossard 2016), and propose a method to measure the effect of quantization error in each DNN layers. Different from (Anwar, Hwang, and Sung 2015; Lin, Talathi, and Annapureddy 2016), which use empirical results to show the relationship between the measurement and DNN accuracy, we conduct a theoretical analysis to show how our proposed method relates to the model accuracy. Furthermore, we show that our bit-width optimization method is more general than the method in (Lin, Talathi, and Annapureddy 2016), which makes our optimization more accurate.

There are also works (Hinton, Vinyals, and Dean 2015; Romero et al. 2014) that use knowledge distillation to train a smaller network using original complex models. It is also possible to combine our quantization framework with knowledge distillation to achieve yet better compression results.

Measuring the effect of quantization noise

In this section, we analyse the effect of quantization on the accuracy of a DNN model. Parameter quantization can result in quantization noise that would affect the performance of the model. Previous works have been done for analyzing the effect of input noise on the DNN model (Fawzi, Moosavi-Dezfooli, and Frossard 2016); here we use this idea to analyse the effect of noise in intermediate feature maps in the DNN model.

Quantization optimization

The goal of our paper is to find a way to achieve optimal quantization result to compress a DNN model. After the quantization, under controlled accuracy penalty, we would like the model size to be as small as possible. Suppose that we have a DNN $\mathcal{F}$ with N layers. Each layer i has s_i parameters, and we apply b_i bit-width quantization in the parameters of layer i to obtain a quantized model $\mathcal{F}'$. Our optimization objective is:

$$\begin{aligned} & min \sum_{i=1}^{N} s_i \cdot b_i \\ & s.t.\ acc_{\mathcal{F}} - acc_{\mathcal{F}'} \leq \Delta_{acc}, \end{aligned} \tag{1}$$

where $acc_{\mathcal{F}}$ is the accuracy of the model $\mathcal{F}$, and Δ_{acc} is the maximum accuracy degradation. Note that it requires enormous computation to calculate the accuracy of the model for all quantization cases. To solve the problem more efficiently, we propose a method to estimate the value of the performance penalty given by $acc_{\mathcal{F}} - acc_{\mathcal{F}'}$.

Quantization noise

Value quantization is a simple yet effective way to compress a model (Han, Mao, and Dally 2015). Here we evaluate the effect of using value quantization on model parameters.

Assume that conducting quantization on a value is equivalent to adding noise to the value:

$$w_q = w + r_w \tag{2}$$

Here w is the original value, $w \in W$, with W the set of weights in a layer. Then, w_q is the quantized value, and r_w is the quantization noise. Assume that we use a uniform quantizer, and that the stepsize of the quantized interval is fixed. Following the uniform quantization analysis in (You 2010), if we consider $\mathbf{r}_w = (r_{w,1}, \cdots, r_{w,N_W})$ as the quantization noise on all weights in W, we have the expectation of $||\mathbf{r}_w||_2^2$ as

$$E(||\mathbf{r}_w||_2^2) = p'_w \cdot e^{-\alpha \cdot b}, \tag{3}$$

where $p'_w = N_W \frac{(w_{min}-w_{max})^2}{12}$, N_W is the number of weights in W, and $\alpha = ln(4)$ (You 2010). Detailed analysis can be found in Supplementary Material. Eq. (3) indicates that every time we reduce the bit-width by 1 bit, $E(||\mathbf{r}_w||_2^2)$ will increase by 4 times. This is equivalent to the quantization efficiency of 6dB/bit in (Gray and Neuhoff 2006).

Measurement for quantization noise

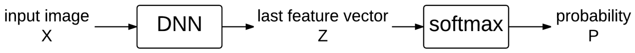


Figure 1: Simple DNN model architecture.

From weight domain to feature domain Eq. (3) shows the quantization noise in the weight domain; here we show how the noise on weight domain can link to the noise in the feature domain.

A simplified DNN classifier architecture is shown in Fig. 1.

Here we define W_i as the weights of layer i in the DNN model $\mathcal{F}$. And Z is the last feature map (vector) of the DNN model $\mathcal{F}$. As we quantize W_i, the quantization noise is $\mathbf{r}_{W_i}$, and there would be a resulting noise $\mathbf{r}_{Z_i}$ on the last feature map Z. Here we define $\mathbf{r}_{Z_i}$ as the noise on last feature map Z that is caused by the quantization only on a single layer i.

As the value of $||\mathbf{r}_{Z_i}||_2^2$ is proportional to the value of $||\mathbf{r}_{W_i}||_2^2$, similar to Eq. (3), the expectation of resulting noise on $\mathbf{r}_{Z_i}$ is:

$$E(||\mathbf{r}_{Z_i}||_2^2) = p_i \cdot e^{-\alpha \cdot b_i} \tag{4}$$

This is proved in later sections, empirical results are shown in Fig. 4.

The effect of quantization noise Similarly to the analysis in (Pang, Du, and Zhu 2017), we can see that the softmax classifier has a linear decision boundary in the last feature vectors Z (Pang, Du, and Zhu 2017) in Fig. 1. The analysis can be found in the Supplementary Material. Then we apply the result of (Fawzi, Moosavi-Dezfooli, and Frossard 2016) to bound the robustness of the classifier with respect to manipulation of weights in different layers.

We define $\mathbf{r}^*$ to be the adversarial noise, which represents the minimum noise to cause misclassification. For a certain input vector $\mathbf{z} = (z_1, \cdots, z_L)$, where L is the number of element in $\mathbf{z}$, the $||\mathbf{r}^*||_2$ is the distance from the datapoint to the decision boundary, which is a fixed value. We define a sorted vector of $\mathbf{z}$ as $\mathbf{z}_{sorted} = (z_{(1)}, \cdots, z_{(L)})$, where the max value is $z_{(1)}$, and second max value is $z_{(2)}$. The result for softmax classifier (or max classifier) can be expressed as: $max(\mathbf{z}) = z_{(1)}$, which is picking up the maximum value in the vector$\mathbf{z}$.

As adversarial noise is the minimum noise that can change the result of a classifier, we can get the adversarial noise for softmax classifier $max(\mathbf{z})$ as $\mathbf{r}^*_{\mathbf{z}_{sorted}} = (\frac{z_{(2)}-z_{(1)}}{2}, \frac{z_{(1)}-z_{(2)}}{2}, 0, \cdots, 0)$, then the norm square of adversarial noise $||\mathbf{r}^*||_2^2 = (z_{(1)} - z_{(2)})^2/2$.

Here we define $\mathbf{r}_Z$ as the noise that we directly add on last feature map Z. We can consider $\mathbf{r}_Z$ as the collective effect of all $\mathbf{r}_{Z_i}$ that caused by the quantization on all layers $i \in \{1, \cdots, N\}$, where N is the number of layers.

As mentioned in (Fawzi, Moosavi-Dezfooli, and Frossard 2016), if we apply random noise $\|\mathbf{r}_Z\|_2^2$ rather than adversarial noise $\|\mathbf{r}^*\|_2$ on the input vector $\mathbf{z}$ for a softmax classifier $max(\mathbf{z})$, it requires higher norm for random noise $\|\mathbf{r}_Z\|_2^2$ to causes prediction error with same probability, compared to adversarial noise $\|\mathbf{r}^*\|_2$.

The following result shows the relationship between the random noise $\|\mathbf{r}_Z\|_2^2$ and adversarial noise $\|\mathbf{r}^*\|_2$, under softmax classifier with a number of classes equal to d:

Lemma 1. *Let $\gamma(\delta) = 5 + 4\ln(1/\delta)$. The following inequalities hold between the norm square of random noise $\|\boldsymbol{r}_Z\|_2^2$ and adversarial noise $\frac{(z_{(1)}-z_{(2)})^2}{2}$.*

$$\frac{\ln d}{d}\gamma(\delta)\|\boldsymbol{r}_Z\|_2^2 \geq \frac{(z_{(1)} - z_{(2)})^2}{2} \tag{5}$$

with probability exceeding $1 - 2\delta$.

The proof of Lemma 1 can be found in the Supplementary Material. The lemma states that if the norm of random noise is $o\left((z_{(1)} - z_{(2)})\sqrt{d/\ln d}\right)$, it does not change the classifier decision with high probability.

Based on Lemma 1, we can rewrite our optimization problem. Assume that we have a model $\mathcal{F}$ with accuracy $acc_{\mathcal{F}}$. After adding random noise $\mathbf{r}_Z$ on the last feature map Z, the model accuracy drops by Δ_{acc}. If we have

$$\theta(\Delta_{acc}) = \frac{d}{\gamma(\frac{\Delta_{acc}}{2acc_{\mathcal{F}}})\ln d}, \tag{6}$$

we have the relation between accuracy degradation and noise $\mathbf{r}_Z$ as:

$$\begin{aligned} &acc_{\mathcal{F}} - acc_{\mathcal{F}'} \leq \Delta_{acc} \Rightarrow \\ &\|\mathbf{r}_Z\|_2^2 < \theta(\Delta_{acc})\frac{(z_{(1)} - z_{(2)})^2}{2} \end{aligned} \tag{7}$$

The detailed analysis can be found in the Supplementary Material. Eq. (7) shows the bound of noise on last feature map Z. However, adding quantization noise to different layers may have different effect on model accuracy. Suppose we have model $\mathcal{F}$ for quantization. By adding noise $\mathbf{r}_{W_i}$

on weights of layer i, we induce the noise $\mathbf{r}_{Z_i}$ on last feature map Z. By quantizing earlier layers, the noise needs to pass through more layers to get $\mathbf{r}_{Z_i}$, which results in a low rank noise $\mathbf{r}_{Z_i}$. For example, when quantizing the first layer, is results in $\mathbf{r}_{Z_1} = (e_1, \cdots, e_{d'}, 0, \cdots, 0)$, and $rank(\mathbf{r}_{Z_1}) = d' < d$. When quantizing the last layer, it results in $\mathbf{r}_{Z_N} = (e_1, \cdots, e_d)$, and $rank(\mathbf{r}_{Z_N}) = d$. In order to let $\mathbf{r}_{Z_N}$ have equivalent effect on model accuracy as $\mathbf{r}_{Z_1}$, $\|\mathbf{r}_{Z_N}\|_2$ should be larger than $\|\mathbf{r}_{Z_1}\|_2$.

By considering the different effects of $\mathbf{r}_{Z_i}$ caused by quantization in different layers, we rewrite Eq. (7) in a more precise form:

$$\begin{aligned} &acc_{\mathcal{F}} - acc_{\mathcal{F}'} = \Delta_{acc} \Rightarrow \\ &\|\mathbf{r}_{Z_i}\|_2^2 = t_i(\Delta_{acc}) \frac{(z_{(1)} - z_{(2)})^2}{2}. \end{aligned} \tag{8}$$

Here $t_i(\Delta_{acc})$ is the robustness parameter of layer i under accuracy degradation Δ_{acc}.

Eq. (8) shows a precise relationship between $\mathbf{r}_{Z_i}$ and Δ_{acc}. If we add quantization noise to layer i of model $\mathcal{F}$, and get noise $\mathbf{r}_{Z_i}$ on last feature map Z, then the model accuracy decreases by Δ_{acc}.

We consider the layer i in model $\mathcal{F}$ as $\mathbf{y}_i = \mathcal{F}_i(\mathbf{y}_{i-1})$, where $\mathbf{y}_i$ is the feature map after layer i. Here we consider that the noise $\mathbf{r}_{y_i}$ would transfer through layers under almost linear transformation (to be discussed in later sections). If we add random noise in the weights of layer i, we have the rank of the resulting noise $\mathbf{r}_{Z_i}$ on last feature map Z given as:

$$rank(\mathbf{r}_{Z_i}) = rank(\prod_{i}^{N} \mathcal{F}_i) \leq min\{rank(\mathcal{F}_i)\} \tag{9}$$

Based on Eq. (9), we have:

$$rank(\mathbf{r}_{Z_1}) \leq \cdots \leq rank(\mathbf{r}_{Z_i}) \leq \cdots \leq rank(\mathbf{r}_{Z_N}) \tag{10}$$

Eq. (10) suggests that the noise on earlier layers of DNN needs to pass through more layers to affect the last feature map Z, the noise $\mathbf{r}_{Z_i}$ on Z would have lower rank, resulting in a lower value of $t_i(\Delta_{acc})$.

From Eq. (8), we can see in particular that when

$$\frac{\|\mathbf{r}_{Z_i}\|_2^2}{t_i(\Delta_{acc})} = \frac{\|\mathbf{r}_{Z_j}\|_2^2}{t_j(\Delta_{acc})}, \tag{11}$$

the quantization on layer i and j have same effect on model accuracy. Based on Eq. (11), $\frac{\|\mathbf{r}_{Z_i}\|_2^2}{t_i(\Delta_{acc})}$ can be a good measurement for estimating the accuracy degradation caused by quantization noise, regardless of which layer to quantize. Consider $\mathbf{x} \in \mathcal{D}$ as the input in dataset $\mathcal{D}$, we have the corresponding feature vector $\mathbf{z} = \mathcal{G}(\mathbf{x}, W)$ in the last feature map Z. By quantizing layer i in model $\mathcal{F}$, we get noise $\mathbf{r}_{\mathbf{z}_i}$ on $\mathbf{z}$. We define the accuracy measurement on layer i as:

$$m_i = \frac{\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}(||\mathbf{r}_{\mathbf{z}_i}||_2^2)}{t_i(\Delta_{acc})} = \frac{\|\mathbf{r}_{Z_i}\|_2^2}{t_i(\Delta_{acc})} \tag{12}$$

The way to calculate $t_i(\Delta_{acc})$ is given by:

$$\begin{aligned} t_i(\Delta_{acc}) = \frac{mean_{\mathbf{r}_{\mathbf{z}_i}}}{mean_{\mathbf{r}^*}} &= \frac{\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}||\mathbf{r}_{\mathbf{z}_i}||_2^2}{\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}}\frac{(z_{(1)}-z_{(2)})^2}{2}} \\ s.t.\ \ &acc_{\mathcal{F}} - acc_{\mathcal{F}'} = \Delta_{acc} \end{aligned} \tag{13}$$

The detailed method to calculate $t_i(\Delta_{acc})$ will be discussed in the experiment section. Note that, based on the optimization result in Eq. (22), the selected value of Δ_{acc} does not matter for the optimization result, as long as the the value of $\frac{t_i(\Delta_{acc})}{t_j(\Delta_{acc})}$ is almost independent w.r.t. Δ_{acc}, which is true according to Fig. 3. So choosing different value of Δ_{acc} does not change the optimization result. In later sections, we use t_i instead of $t_i(\Delta_{acc})$ for simplicity.

From Eq. (12), based on the linearity and additivity of the proposed estimation method (shown in later sections), the measurement of the effect of quantization error in all the layers of the DNN model is shown in Eq. (20).

After we define the accuracy measurement for each layer of model, based on Eq. (8), we can then rewrite the optimization in Eq. (1) as

$$\begin{aligned} min \sum_{i=1}^{N} s_i \cdot b_i \\ s.t.\ m_{all} = \sum_{i=1}^{N} m_i \leq C, \end{aligned} \tag{14}$$

where m_{all} is the accuracy measurement for all layers, and C is a constant related to model accuracy degradation Δ_{acc}, with higher C indicating higher Δ_{acc}.

Linearity of the measurements

In this section we will show that the DNN model are locally linear to the quantization noise measurement $||\mathbf{r}_W||_2$, under the assumption that the quantization noise is much smaller than the original value: $||r_w||_2 \ll ||w||_2$. That is, if a quantization noise $||\mathbf{r}_{W_i}||_2$ on layer i leads to $||\mathbf{r}_{Z_i}||_2$ on last feature vector Z, then we have a quantization noise $\alpha \cdot ||\mathbf{r}_{W_i}||_2$ on layer i leads to $\alpha \cdot ||\mathbf{r}_{Z_i}||_2$ on last feature vector Z.

For linear layers like convolutional layers and fully connected layers in the DNN model, the linearity for noise is obvious. Here we mainly focus on the non-linear layers in the DNN model, such as ReLU and Max-pooling layers.

ReLU layers The ReLU layers is widely used to provide nonlinear activation for DNN. Given the input $a \in A$ to a ReLU layer, the output value $z \in Z$ is calculated as:

$$z = ReLU(a) = \begin{cases} a, & if\ a > 0 \\ 0, & if\ a <= 0 \end{cases} \tag{15}$$

From Eq. (15) we can see that the ReLU layer is linear to noise in most cases. The non-linearity happens only when the noise r_w crossing the zero point, which has small probability when the noise is sufficiently small.

Max-pooling layers Max-pooling is a nonlinear downsampling layer that reduces the input dimension and controls overfitting. We can consider that max-pooling acts as a max filter to the feature maps.

Similarly to the ReLU layer, which can be described as $z = ReLU(a) = max(0, a)$, the max-pooling layer can be describes as $z = maxpool(\{a_i\}) = max(\{a_i\})$, where $i = 1, \cdots, P$, with P the kernel size for max-pooling. The linearity for noise holds when the noises are sufficiently small and do not alter the order for $\{a_i\}$.

Other layers For other non-linear layers like Sigmoid and PReLU, the linearity for small noises still holds under the assumptions that the function is smooth along most input ranges, and the noise has very low probability to cross the non-linear region.

Based on the linearity assumption, as we model the quantization noise on weight as Eq. (3), the resulting noise on last feature vector Z can be modeled as:

$$||\mathbf{r}_{Z_i}||_2^2 = p_i \cdot e^{-\alpha \cdot b_i} \tag{16}$$

Additivity of the measurements

Noise on single multiplication Pairwise multiplication is a basic operation in the convolutional layers and fully connected layers of DNN. Given one value in the input $a \in A$, one value in the weight matrix $w \in W$, we have the pairwise multiplication as $a \cdot w$. If we consider noise in both input $a \in A$ and $w \in W$, we have noised value $a_q \in A_q$ and $w_q \in W_q$, and finally $a_q \cdot w_q = (a + r_a) \cdot (w + r_w)$.

Noise on one layer Given a convolutional layer input A with size $M \times N \times C$, conv kernel K with size $M_k \times N_k \times C \times D$, and stride size s, we have the output feature map Z with size $(M/s) \times (N/s) \times D$. Here M and N are the height and width of input, C is the number of channels of input. M_k and N_k are the height and width of the conv kernel, D is the depth of output feature map.

The analysis on fully connected layers will be similar to the analysis on convolutional layers. It can be considered as a special case of convolutional layers when M, N, M_k, and N_k are equal to 1. For a single value $z_{p,q,d} \in Z$, the noise term of $z_{p,q,d}$ can be expressed as:

$$\begin{aligned} r_{z_{p,q,d}} &= \sum_{i=1}^{M_k \cdot N_k \cdot C} (a_i \cdot r_{w_{i,d}} + r_{a_i} \cdot w_{i,d} + r_{a_i} \cdot r_{w_{i,d}}) + r_{b_d} \\ &\approx \sum_{i=1}^{M_k \cdot N_k \cdot C} (a_i \cdot r_{w_{i,d}} + r_{a_i} \cdot w_{i,d}) \end{aligned} \tag{17}$$

The calculation details can be found in Supplementary Material. Note that the term r_{b_d} can be ignored under the assumption that w and b have same bit-width quantization. The term $r_{a_i} \cdot r_{w_{i,d}}$ can be ignored under the assumption that $||r_a||_2 \ll ||a||_2$ and $||r_w||_2 \ll ||w||_2$.

From Eq. (17) we can see that: 1) adding noise to input feature maps and weights separately and independently, is equivalent to adding noise to both input feature maps and weights; 2) regarding the output feature map $\mathbf{z} \in Z$, adding noise to the input feature maps and weights and doing layer operation (pairwise product), is equivalent to adding the noise directly to the output feature map. We will use these two properties in later sections.

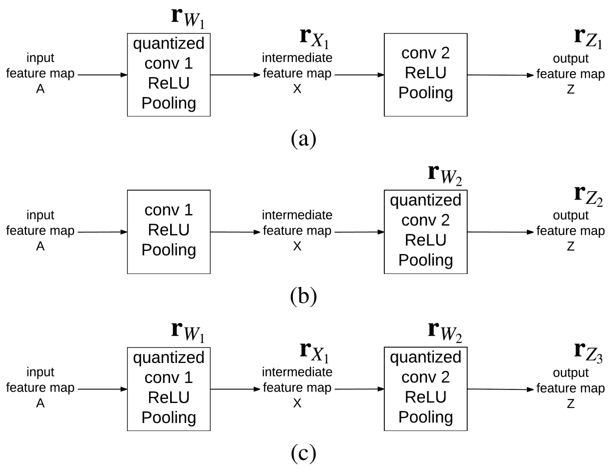


Figure 2: Effect of adding noise to multiple layers.

Adding noise to multiple layers Fig. 2 shows a 2-layer module inside a DNN model. Given input feature map A, after the first conv layer, an intermediate feature map X is generated, then after the second conv layer, output feature map Z is generated. Fig. 2(a) and 2(b) show the effect of noise on layer 1 and 2, respectively. And Fig. 2(c) shows the effect of noise on both layer 1 and 2. By analysing the additivity of $\|\mathbf{r}_Z\|_2^2$, we have:

$$||\mathbf{r}_{\mathbf{z}_3}||_2^2 \doteq ||\mathbf{r}_{\mathbf{z}_1}||_2^2 + ||\mathbf{r}_{\mathbf{z}_2}||_2^2 \tag{18}$$

Detailed analysis can be found in Supplementary Material. Eq. (18) holds under the assumption that $\mathbf{r}_{\mathbf{z}_1}$ and $\mathbf{r}_{\mathbf{z}_2}$ are independent. This is reasonable in our case, as $\mathbf{r}_{\mathbf{z}_1}$ and $\mathbf{r}_{\mathbf{z}_2}$ are caused by $\mathbf{r}_{W_1}$ and $\mathbf{r}_{W_2}$ which are two independent quantization noises. This independence between $\mathbf{r}_{\mathbf{z}_1}$ and $\mathbf{r}_{\mathbf{z}_2}$ is also important for our proposed estimation method.

We can extend Eq. (18) to the situation of N layers:

$$||\mathbf{r}_{\mathbf{z}}||_2^2 = \sum_{i=1}^{N} ||\mathbf{r}_{\mathbf{z}_i}||_2^2 \tag{19}$$

If we consider the linearity and additivity of the proposed measurement, from Eq. (12) and Eq. (19), as well as the independence of the measurement among different layers, we have the measurement of the effect of quantization errors in all layers in DNN model:

$$m_{all} = \sum_{i=1}^{N} m_i = \sum_{i=1}^{N} \frac{||\mathbf{r}_{Z_i}||_2^2}{t_i} \tag{20}$$

Eq. (20) suggests that the noise effect of adding noise to each layer separately and independently, is equivalent to the effect of adding noise to all layers simultaneously. We use

Eq. (12) as the measurement for noise effect on layer i, and the effect of adding noise to all layers can be predicted using Eq. (20).

Layer-wise bit-width optimization

In this section we show the approach for optimizing the layer-wise bit-width quantization to achieve an optimal compression ratio under certain accuracy loss.

Following the discussion from the optimization problem in Eq. (14), our goal is to constraint Eq. (20) to be a small value while minimizing the model size.

Adaptive quantization on multiple layers

Based on Eq. (16) and (20), the optimization Eq. (14) can be expressed as:

$$\begin{aligned} & min \sum_{i=1}^{N} s_i \cdot b_i \\ & s.t. \sum_{i=1}^{N} \frac{p_i}{t_i} \cdot e^{-\alpha \cdot b_i} \leq C \end{aligned} \tag{21}$$

The optimal value of Eq. (21) can be reached when:

$$\frac{p_1 \cdot e^{-\alpha \cdot b_1}}{t_1 \cdot s_1} = \frac{p_2 \cdot e^{-\alpha \cdot b_2}}{t_2 \cdot s_2} = \cdots = \frac{p_N \cdot e^{-\alpha \cdot b_N}}{t_N \cdot s_N} \tag{22}$$

The detailed analysis can be found in Supplementary Material.

Optimal bit-width for each layer

From Eq. (22) we can directly find the optimal bit-width for each layer using the following procedure:

- Calculate t_i (Eq. (13)):
  - First, calculate the mean value of adversarial noise for the dataset: $mean_{\mathbf{r}^*} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{(z_{(1)} - z_{(2)})^2}{2}$.
  - Then, fix Δ_{acc} value. For example, $\Delta_{acc} = 10\%$. Note that the selection of Δ_{acc} value does not affect the optimization result.
  - For each layer i, change the amount of noise $\mathbf{r}_{W_i}$ added in weight W_i, until the accuracy degradation equals to Δ_{acc}. Then, record the mean value of noise $\mathbf{r}_{\mathbf{z}_i}$ on the last feature map Z: $mean_{\mathbf{r}_{\mathbf{z}_i}} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} ||\mathbf{r}_{\mathbf{z}_i}||_2^2$.
  - The value t_i can be calculated as: $t_i(\Delta_{acc}) = \frac{mean_{\mathbf{r}_{\mathbf{z}_i}}}{mean_{\mathbf{r}^*}}$.
  - The details for the calculation of t_i can be found in Fig. 3.
- Calculate p_i:
  - First, for each layer i, fix b_i value. For example, use $b_i = 10$.
  - Then, record the mean value of noise $\mathbf{r}_{\mathbf{z}_i}$ on the last feature map Z: $mean_{\mathbf{r}_{\mathbf{z}_i}} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} ||\mathbf{r}_{\mathbf{z}_i}||_2^2$.
  - The value p_i can be calculated using Eq. (16): $mean_{\mathbf{r}_{\mathbf{z}_i}} = ||\mathbf{r}_{Z_i}||_2^2 = p_i \cdot e^{-\alpha \cdot b_i}$.
- Calculate b_i:
  - Fix the bitwidth for first layer b_1, for example, $b_1 = 10$. Then bitwidth for layer i can be calculated using the Eq. (22): $\frac{p_1 \cdot e^{-\alpha \cdot b_1}}{t_1 \cdot s_1} = \frac{p_i \cdot e^{-\alpha \cdot b_i}}{t_i \cdot s_i}$

The detailed algorithm about the above procedure can be found in Supplementary Material. Note that, by selecting different b_1, we achieve different quantization result. A lower value of b_1 results in higher compression rate, as well as higher accuracy degradation.

Comparison with SQNR-based approach

Based on the SQNR-based approach (Lin, Talathi, and Annapureddy 2016), the optimal bit-width is reached when:

$$\frac{e^{-\alpha \cdot b_1}}{s_1} = \frac{e^{-\alpha \cdot b_2}}{s_2} = \cdots = \frac{e^{-\alpha \cdot b_N}}{s_N} \tag{23}$$

The proof can be found in Supplementary Material. Note that compared with our result in Eq. (22), the parameters p_i and t_i are missing. This is consistent with the assumption of the SQNR-based approach, where two layers having the same bit-width for quantization would have the same SQNR value; hence the effects on accuracy are equal. This makes the SQNR-based approach a special case of our approach, when all layers in the DNN model have the equal effect on model accuracy under the same bit-width.

Experimental results

In this section we show empirical results that validate our assumptions in previous sections, and evaluate the proposed bit-width optimization approach.

All codes are implemented using MatConvNet (Vedaldi and Lenc 2015). All experiments are conducted using a Dell workstation with E5-2630 CPU and Titan X Pascal GPU.

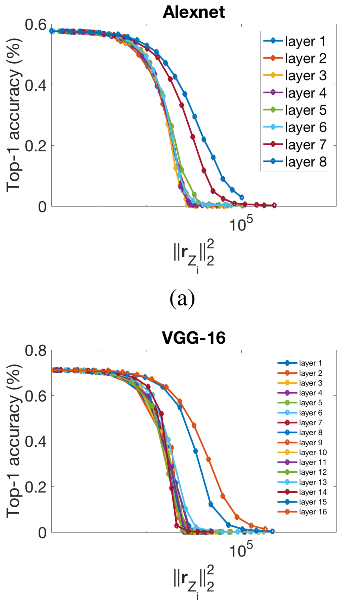
Empirical results about measurements

To validate the effectiveness of the proposed accuracy estimation method, we conduct several experiments. These experiments validate the relationship between the estimated accuracy, the linearity of the measurement, and the additivity of the measurement.

Here we use Alexnet (Krizhevsky, Sutskever, and Hinton 2012), VGG-16 (Simonyan and Zisserman 2014), GoogleNet (Szegedy et al. 2015), and Resnet (He et al. 2016) as the model for quantization. Each layer of the model is quantized separately using uniform quantization, but possibly with different bit-width. The quantized model is then tested on the validation set of Imagenet (Krizhevsky, Sutskever, and Hinton 2012), which contains 50000 images in 1000 classes.

Calculate t_i As Eq. (12) is proposed to measure the robustness of each layer, we conduct an experiment to find t_i value. We use Alexnet as an example.

First, we calculate the adversarial noise for Alexnet on the last feature vector Z. The calculation is based on Eq. (13). The mean value of $||\mathbf{r}^*||_2^2$ for Alexnet is $mean_{\mathbf{r}^*} = 5.33$.



(a)

(b)

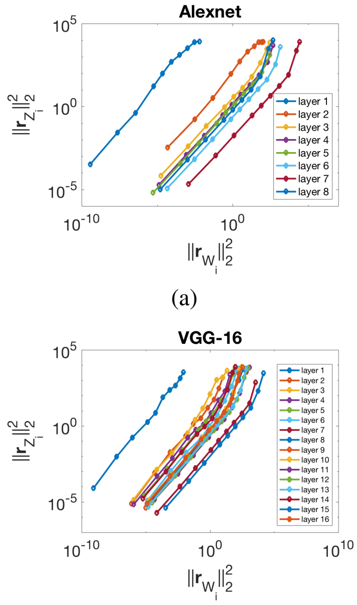
Figure 3: The relationship between different $||\mathbf{r}_{Z_i}||_2^2$ and model accuracy.

The distribution of $||\mathbf{r}^*||_2^2$ for Alexnet on Imagenet validation set can be found in Supplementary Material.

After finding $||\mathbf{r}^*||_2^2$ value, the value of t_i is calculated based on Fig. 3(a) and Eq. (13). We set the accuracy degradation to be roughly half of original accuracy (57%), which is 28%. Based on the values in Fig. 3(a), t_1 to t_6 are equal to 5.2×10^2, $t_7 = 1.3 \times 10^3$, and $t_8 = 2.0 \times 10^3$.

Here we show the example for Alexnet of how to calculate the t_i value. Note that for other networks like VGG-16, GoogleNet, and Resnet, we also observe that only the t_i value for the last 1 or 2 layers are obviously different than the other t_i values. During our calculation, we can thus focus on the t_i values for the last several layers. Furthermore, in Fig. 3(a), we find the $||\mathbf{r}_Z||_2^2 - accuracy$ relationship for different amounts of noise, which requires a lot of calculations. In real cases, we use binary search to find appropriate points under the same accuracy degradation. This makes the process to calculate t_i fast and efficient. Typically, for a deep model with N layers, and dataset with $|\mathcal{D}|$ size, we require $O(\tau N|\mathcal{D}|)$ forward passes to calculate accuracy. Here τ is the trial times over one layer. We can reduce it to $O(\tau N'|\mathcal{D}|)$(with $N' << N$) by only calculating t_i values for the last N' layers.

In our experiments, the calculation of t_i is the most time-consuming part of our algorithm. We use around 15 mins to calculate the t_i value for Alexnet (30 sec for forward pass on the whole dataset), and around 6 hours to calculate the t_i value for Resnet-50 (2 min for forward pass on the whole dataset). This time can be reduced if we only calculate t_i values for the last few layers.



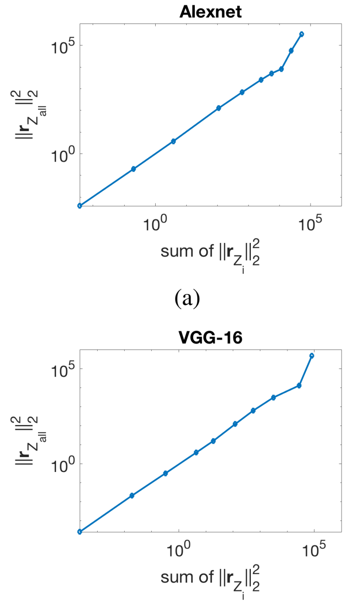
(a)

(b)

Figure 4: The relationship between $||\mathbf{r}_{W_i}||_2^2$ and $||\mathbf{r}_{Z_i}||_2^2$.

Linearity of measurements Fig. 4 shows the relationship between the norm square of noise on quantized weight $||\mathbf{r}_{W_i}||_2^2$ and $||\mathbf{r}_{Z_i}||_2^2$ on different layers. When the quantization noise on weight is small, we can observe linear relationships. While it is interesting to see that, when the quantization noise is large, the curve does not follow exact linearity, and curves for earlier layers are not as linear as later layers. One possible explanation is that earlier layers in a DNN model are affected by more non-linear layers, such as ReLU and Max-pooling layers. When the noise is large enough to reach the non-linear part of the layer functions (i.e. the zero point of the ReLU function), the curves become non-linear. It is worth noting that, when the non-linearity in most layers happens, the accuracy of the model is already heavily affected (become near zero). So this non-linearity would not affect our quantization optimization process.

Additivity of measurements Fig. 5 shows the relationship between $\sum_i^N ||\mathbf{r}_{Z_i}||_2^2$ when we quantize each layer separately, and the value $||\mathbf{r}_Z||_2^2$ when we quantize all layers together. We can see that when the quantization noise is small, the result closely follows our analysis that $||\mathbf{r}_Z||_2^2 = ||\mathbf{r}_{Z_i}||_2^2$; it validates the additivity of $||\mathbf{r}_Z||_2^2$. When the quantization noise is large, the additivity of $||\mathbf{r}_Z||_2^2$ is not accurate. This result fits our assumption in Eq. (17), where the additivity holds under the condition $||\mathbf{r}_{W_i}||_2 \ll ||W_i||_2$ for all layer i in the DNN model. When the noise is too high and we observe the inaccuracy of additivity, the model accuracy is already heavily degraded (near zero). Hence it does not affect the quantization optimization process which rather works in low noise regime.



(a)

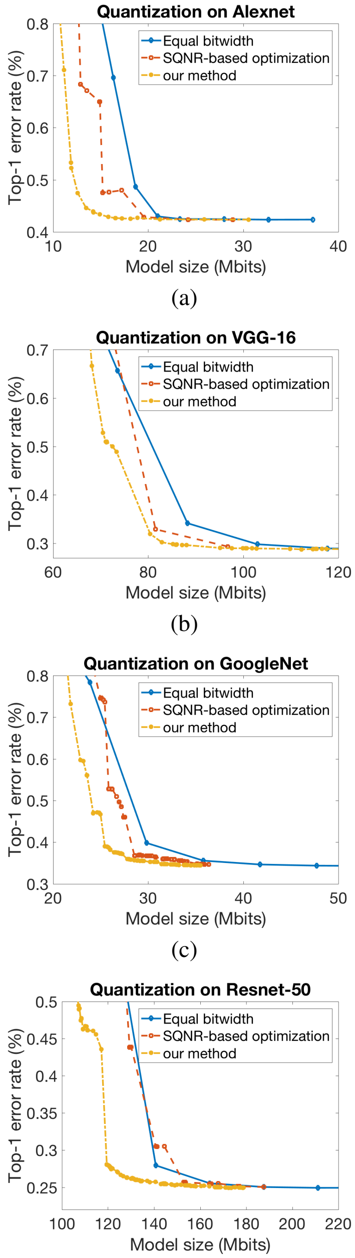
(b)

Figure 5: The value of $\sum_i^N ||\mathbf{r}_{Z_i}||_2^2$ when quantize each layer separately, compare to $||\mathbf{r}_Z||_2^2$ when quantize all layers simultaneously.

Optimal bit-width for models

After the validation of the proposed measurement, we conduct experiments to show the results on adaptive quantization. Here we use Alexnet (Krizhevsky, Sutskever, and Hinton 2012), VGG-16 (Simonyan and Zisserman 2014), GoogleNet (Szegedy et al. 2015), and Resnet-50 (He et al. 2016) to test our bit-width optimization approach. Similarly to the last experiments, the validation set of Imagenet is used. As the SQNR-based method (Lin, Talathi, and Annapureddy 2016) only works for convolutional layers, here we keep the fully connected layers with 16 bits .

Fig. 6 shows the quantization results using our method, SQNR-based method (Lin, Talathi, and Annapureddy 2016), and equal bit-width quantization. The equal bit-width quantization means that the number of quantization intervals in all layers are the same. For all three methods, we use uniform quantization for each layer. We can see that for all networks, our proposed method outperforms SQNR-based method, and achieves smaller model size for the same accuracy degradation. It is interesting to see that the SQNR-based method does not obviously outperform equal quantization on the Resnet-50 model. One possible reason is that Resnet-50 contains 1×1 convolutional layers in its ”bottleneck” structure, which is similar to fully connected layers. As the authors claim in (Lin, Talathi, and Annapureddy 2016), the SQNR-based method does not work for fully connected layers. Note that our method generates more datapoints on the figure, because the optimal bit-width for different layers may contain different decimals. And by rounding the optimal bit-width in different ways, we can generate more bit-width combinations than the SQNR-based methods.



(a)

(b)

(c)

(d)

Figure 6: Model size after quantization, v.s. accuracy. To compare with SQNR-based method (Lin, Talathi, and Annapureddy 2016), only convolutional layers are quantized.

The results for quantization on all layers are shown in Supplementary Material. For Alexnet and VGG-16 model, our method achieves $30 - 40\%$ smaller model size with the same accuracy degradation, while for GoogleNet and Resnet-50, our method achieves $15 - 20\%$ smaller model size with the same accuracy degradation. These results indicate that our proposed quantization method works better for models with more diverse layer size and structures, like

Alexnet and VGG.

Conclusions

Parameter quantization is an important process to reduce the computation and memory costs of DNNs, and to deploy complex DNNs on mobile equipments. In this work, we propose an efficient approach to optimize layer-wise bit-width for parameter quantization. We propose a method that relates quantization to model accuracy, and theoretically analyses this method. We show that the proposed approach is more general and accurate than previous quantization optimization approaches. Experimental results show that our method outperforms previous works, and achieves $20-40\%$ higher compression rate than SQNR-based methods and equal bit-width quantization. For future works, we will consider combining our method with fine-tuning and other model compression methods to achieve better model compression results.

References

Anwar, S.; Hwang, K.; and Sung, W. 2015. Fixed point optimization of deep convolutional neural networks for object recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1131–1135.

Deng, L.; Hinton, G.; and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8599–8603.

Do, T.-T.; Doan, A.-D.; and Cheung, N.-M. 2016. Learning to hash with binary deep neural network. In *European Conference on Computer Vision (ECCV)*, 219–234. Springer.

Fawzi, A.; Moosavi-Dezfooli, S.-M.; and Frossard, P. 2016. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems (NIPS)*. 1632–1640.

Figurnov, M.; Ibraimova, A.; Vetrov, D. P.; and Kohli, P. 2016. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems (NIPS)*, 947–955.

Gray, R. M., and Neuhoff, D. L. 2006. Quantization. *IEEE Transactions on Information Theory (TIT)* 44(6):2325–2383.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 1737–1746.

Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M. A.; and Dally, W. J. 2016. Eie: efficient inference engine on compressed deep neural network. In *Proceedings of the IEEE International Symposium on Computer Architecture (ISCA)*, 243–254.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hoang, T.; Do, T.-T.; Tan, D.-K. L.; and Cheung, N.-M. 2017. Selective deep convolutional features for image retrieval. *arXiv preprint arXiv:1707.00809*.

Hwang, K., and Sung, W. 2014. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, 1–6.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, 1097–1105.

Lin, D.; Talathi, S.; and Annapureddy, S. 2016. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning (ICML)*, 2849–2858.

Pang, T.; Du, C.; and Zhu, J. 2017. Robust deep learning via reverse cross-entropy training and thresholding test. *arXiv preprint arXiv:1706.00633*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Sun, F.; Lin, J.; and Wang, Z. 2016. Intra-layer nonuniform quantization of convolutional neural network. In *2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*, 1–5.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 1–9.

Vedaldi, A., and Lenc, K. 2015. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*.

Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; and Cheng, J. 2016. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4820–4828.

You, Y. 2010. *Audio Coding: Theory and Applications*. Springer Science & Business Media.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision (ECCV)*, 818–833. Springer.

Zhou, Y.; Do, T. T.; Zheng, H.; Cheung, N. M.; and Fang, L. 2016. Computation and memory efficient image segmentation. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* PP(99):1–1.