Convolution without Multiplication: A General Speed Up Strategy for CNN

Anonymous CVPR submission

Paper ID 2175

Abstract

Convolutional Neural Networks (CNN) have achieved great success in many computer vision tasks. Due to the fact that most existing CNN models are computationally expensive, it is difficult to deploy these models on mobile or other low-cost devices with limited power budgets. Therefore, CNN model compression and acceleration becomes a hot research topic in deep learning area. Typical schemes including parameter pruning and sharing, low-rank factorization, compact convolutional filters and knowledge distillation. The purpose of these methods is to speed up the algorithm without significantly decreasing the accuracy. To this end, we propose a general accelerate scheme, where the floating point multiplication is replaced by integer addition. The motivation is based on the fact that every floating point can be replaced by the summation of a exponential series. Therefore the multiplication between two floating points can be converted to the addition among exponential. In the section of experiments, we directly apply the proposed scheme to AlexNet, VGG, ResNet for image classification, and Faster-RCNN for object detction. The results acquired from ImageNet and PASCAL VOC show that the proposed quantized scheme has promising performance, even with only one item of exponential. We also analyze the efficiency on main stream FPGA, which show the proposed quantized scheme can achieve more than 20× acceleration with only slight accuracy loss.

1. Introduction

Deep neural networks are becoming the preferred tool in many machine learning and computer vision applications. However, with networks going deeper, the number of parameters in a network becomes larger. Typically, mainstream deep networks always have millions or even billions of parameters, which make deploying a deep network on a small device more difficult than before. On the other hand, there is an increasing demand for deploying deep CNN models in cell phone and other wearable devices such as FPGA. Therefore, reduce the storage and the computational cost of CNN becomes a critical task.

In recent years, much work has been done to compress, or to accelerate the feed forward operations in CNNs. In [2], Cheng et. al. classified these approaches into four categories. Which are respectively as parameter pruning and sharing, low-rank factorization, transfer/compact convolutional filters, and knowledge distillation. Parameter pruning and sharing focus on remove redundant and unimportant parameters. Low-rank factorization employed matrix decomposition to measure the informative degree of each parameter. The compact convolutional filters aim at constructing efficient structural filters to reduce the storage. The knowledge distillation methods try to learn a distilled model to reproduce the output of a large network.

Although these methods have achieved great progress in the past years, each scheme still has some issues that are hard to solve. For example, the accuracy of binary or quantized nets may decay when dealing with large CNNs. Pruning scheme require manual setup of sensitivity for layers. Low-rank factorization needs approximation layer by layer, and thus is difficult to perform global parameter compression. Transfer/compact approaches are suitable for wide/flat models rather than narrow/special architectures. Knowledge distillation can only be applied to classification tasks with softmax loss function. In other words, general and simple speed up scheme with low accuracy loss is still an challenging task.

To this end, this paper introduce a novel speed up strategy that can be directly applied in each CNN with floating point multiplication. The scheme is simple, efficient without any training process. The basic idea is to reduce the number of floating point operation in CNN. Specifically, we aims at finding an alternative scheme that using integer addition to replace floating point multiplication. We achieve such a target by replacing each floating point with a exponential series. Then the multiplication between two floating points is converted to the addition among exponential. We demonstrate that our quantized neural networks are comparable to standard full precision networks while requiring less memory and significantly reduce floating point multiplication.

The contribution of this paper is two-fold: First, we introduce a novel way of parameter quantization to accelerate the feed foward operations of CNNs and show the results are comparable with original CNN models. Second, since most floating points in the network are replaced by short integers, the storage is also been reduced. To summarize, compared to each original CNN model, its quantized CNN model has faster feed forward speed, lower storage demand with almost no accuracy loss.

2. Related Works

There has been a significant amount of work on accelerating and compressing CNNs. In order to point out the main contributions of the proposed quantized scheme, we first give a brief introduction of main stream accelerate schemes. Then we focus on analyzing the related works of quantization and binarization, which are mostly related to our method.

Firstly, low rank factorization and sparsity use matrix decomposition to evaluate the informative degree of each parameters in CNNs. Typical methods including separable 1D filters[2], low-rank approximation and clustering schemes[3], tensor decomposition schemes[4], which has 4.5× speedup with only 1% drop in accuracy, Canonical Polyadic (CP) decomposition[5], and low-rank tensor decomposition for training low-rank constrained CNNs[6], etc. Generally speaking, low-rank approaches are straightforward for model compression and acceleration.

Secondly, as for the compact convolutional filters, the basic idea is motivated by recent works in [2], which introduced the equivariant group theory. Following this trend, there are many works been proposed to build a convolutional layer from a set of base filters[32]-[17]. These methods can achieve competitive performance for wide/flat architectures but not narrow/special ones. On the other hand, compact filter for convolution can directly reduce the computation cost, such as decomposing 3×3 convolution into two 1×1 convolutions[25]. Similarly, SqueezeNet [29] proposed a scheme to replace 3×3 convolution with 1×1 convolution. Take AlexNet for example, SqueezeNet created a compact network with about 50 fewer parameters.

Thirdly, as for the parameter pruning and sharing scheme, a recent fashion trend is to prune redundant and non-informative weights in a pre-trained CNN model. Such as data-free pruning to remove redundant neurons[24], optimize the total number of parameters and operations in the entire network[10], HashedNets[1], soft weight-sharing [27], group sparsity constraint[14], structured sparsity regularizer [29], group-sparse regularizer[33], select and prune unimportant filters pruning based on L1-norm [16]. Note that most pruning criteria require manual setup of sensitivity for layers, which demands fine-tuning of the parameters and could be cumbersome for some application.

Finally, the quantization and binarization schemes aim at compressing the original network by reducing the number of bits required to represent each weight. This strategy is the most related to our approach. Among all the network quantization algorithms, clustering-based scalar quantization is the most widely used[7][31]. In [28], the floating point is been converted to 8-bit quantization, which results in significant speed-up with slight loss of accuracy. Similarly, [8] proposed a 16-bit fixed-point representation based CNN training, which significantly reduced memory usage and floating point operations. In particular, recent trends show that the combination of different schemes achieved state-of-the-art performance, such as the method proposed in [9]. The algorithm first pruned unimportant connections and only retrained sparse connections. Consequently, weight sharing based on Huffman coding is used to the quantized weights. The network is then re-trained with remaining sparse connections. As for the binarization scheme, there are also many works that directly train CNNs with binary weights. Typical works include BinaryConnect [5], BinaryNet [4] and XNOR Networks [19]. The main purpose is to directly learn binary weights during the training process. In addition, [18] showed that networks trained with back propogation are robust against specific weight distortions, including binary weights. one of the disadvantage is that the accuracy of binarization network always significantly lower than its original model, especially targeting with large CNNs.

3. The Motivation

In order to make things clearer, we'll show our motivation from two experiments. We found a fact that slight changes of the parameters in a deep convolutional neural networkmay not influence the discriminative ability of features extracted via the network.

The first experiment focuses on evaluating the differences between original CNN and rectified CNN. A rectified CNN is generated from a typical CNN via a quantification process. Particularly, we retain several different digits after the decimal point of the convolutional parameters during the process of parameter quantification in original CNN. As shown in Table 1, we test three widely used CNN models for image classification in ImageNet. In the first row, the results of original CNNs are regarded as the baseline, including AlexNet[13], VGG[23], and ResNet[11]. On the other hand, the rest 10 rows show the classification results that obtained from the quantized CNN. Specifically, the bold numbers numbers denoted the results that are identical with baseline, where the red numbers show that the quantized parameters perform better than baseline. On can see that when we hold two digits after the decimal point of the convolutional parameters, the results are very close to original CNNs. In addition, if we only save two digits after the decimal point, the results are almost identical with the original CNNs. Specially, in ResNet50, the quantized results are even better than the original network. Therefore, we can remark this phenomenon as follow:

Table 1.The performance of digit-quantized models for image classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	70.956	89.9161	75.102	92.2002	56.8	79.946
1-digit	0.1	0.526	0.1	0.5	0.114	0.522
2-digits	69.354	88.908	73.6499	91.3502	56.202	79.636
3-digits	70.918	89.8981	**75.202**	**92.2122**	56.704	79.904
4-digits	70.948	**89.9261**	**75.224**	92.1782	**56.802**	**79.958**
5-digits	70.952	**89.9161**	**75.212**	92.1662	**56.802**	**79.946**
6-digits	**70.956**	**89.9161**	**75.218**	92.1662	**56.8**	**79.946**
7-digits	**70.956**	**89.9161**	**75.218**	92.1682	**56.8**	**79.946**
8-digits	**70.956**	**89.9161**	**75.216**	92.1682	**56.8**	**79.946**
9-digits	**70.956**	**89.9161**	**75.216**	92.1682	**56.8**	**79.946**
10-digits	**70.956**	**89.9161**	**75.216**	92.1682	**56.8**	**79.946**

Remark 1: Slight changes of the CNN parameters do not severely influence the performance. Specially, only three digits after the decimal point are important in CNNs. Namely, the digits after the first three digits in each parameter is not essential.

According to remark 1, slight changes of the parameters in a convolutional template do not influence the performance of CNN. Since the main purpose of our method is to convert the multiplication into addition, we then try to convert the convulutional parameters into the accumulations of exponential series. Theorem 1 explains that the combination can approximate any floating point with arbitrary precision.

Theorem 1: *Given a floating point C in [-1, 1] and 0<a≤ 2, for any ε>0, there exists an integer K, a negative integer serial* x_i *(i=0,-1,-2,...), and a binary varaiable* $\delta_i \in \{-1,+1\}$ *, which satisfying the following equation:*

$$|C - \sum_{i=1}^{K} \delta_i a^{x_i}| \le \varepsilon \quad (1)$$

Proof. We use a strategy of greedy search to proof this theorem . Without loss of generality, let $C>0$. According to the definition of C, there exists a negative integer $m_1 < 0$ that satisfies $a^{m_1} \le C \le a^{m_1+1}$. Let

$$x_1 = \begin{cases} m_1, & |C - a^{m_1}| \le |C - a^{m_1+1}| \\ m_1+1, & \text{otherwise} \end{cases},$$

then we have

$$C = \begin{cases} a^{x_1} + r_1, x_1 = m_1 \\ a^{x_1} - r_1, x_1 = m_1+1 \end{cases}.$$

The residual is denoted as $r_1 = \min\{|C - a^{m_1}|, |C - a^{m_1+1}|\}$. Likewise, there exists a negative integer m_2 that satisfying $a^{m_2} < r_1 < a^{m_2+1}$. Also the residual can be derived as $r_2 = \min\{|r_1 - a^{m_2}|, |r_1 - a^{m_2+1}|\}$. The rest can be done in the same manner. Namely, we can deduce a residual sequence r_1, r_2, ... r_n, ... Note that the residual r_i is equal to r_{i-1} subtracts a positive number. As a consequent, r_1, r_2, ... r_n, ... is a monotonically decreasing sequence.

On the other hand, given a≤2, we have $r_i \le \frac{a^{m_i+1} - a^{m_i}}{2} \le \frac{a^{m_i}(a-1)}{2} \le \frac{a^{m_i}}{2}$, which means that each r_i falls in different intervals. In other words, with the increase of m, r_i converge to 0. Therefore, for any ε>0, there exists an integer K, for any $i \ge K$, we have $r_i \le \varepsilon$. According to the definition of r_i and x_i, the residual sequence $x_1, x_2, \ldots x_n, \ldots$ satisfies $|C - \sum_{i=1}^{K} \delta_i a^{x_i}| \le \varepsilon$. ■

Theorem 1 shows that multiplication of floating points can be converted into exponential (typically integer) addition, with very slight accuracy loss. Recall the first experiment, we show that when each convolutional parameter is replaced by an approximated floating point, the result is almost identical. Consequently, we'll explain how to use exponential addition to construct a float point approximating to the original parameter.

As shown in Figure 1, let x_i in {0,-1,..., -N}, item denotes the number of components that construct a floating point; the vertical axis presents the residual/error which is the difference between the original floating point and its approximated combination, which is acquired from the greedy search strategy; the horizontal axis denotes the range of input floating point, where the interval [0,1] is quantized as {0,1,..., 255}; $\varepsilon = a^{-N}$ stands for the right endpoints of the first interval, which is closely related to the minimal error/residual. It is worth noting that smaller ε means smaller error/residual, where $N = \log_a \varepsilon$ indicates that smaller ε may results in larger look up table.

From Figure 1(a) to 1(d), one can see a common phenomenon is that with the increase of the number of items, the error will be drastically decreased. In particular, when item=3, the fitting error is very small. This phenomenon reveals that any floating point can be fitting by a combination of exponential, given any accuracy. On the other hand, with the decrease of ε is accompanied by the increase of the range of N . In this situation, there exist more smaller intervals/values for fitting, which may decrease the fitting error of exponential combination. Therefore, we can remark this phenomenon as follow:

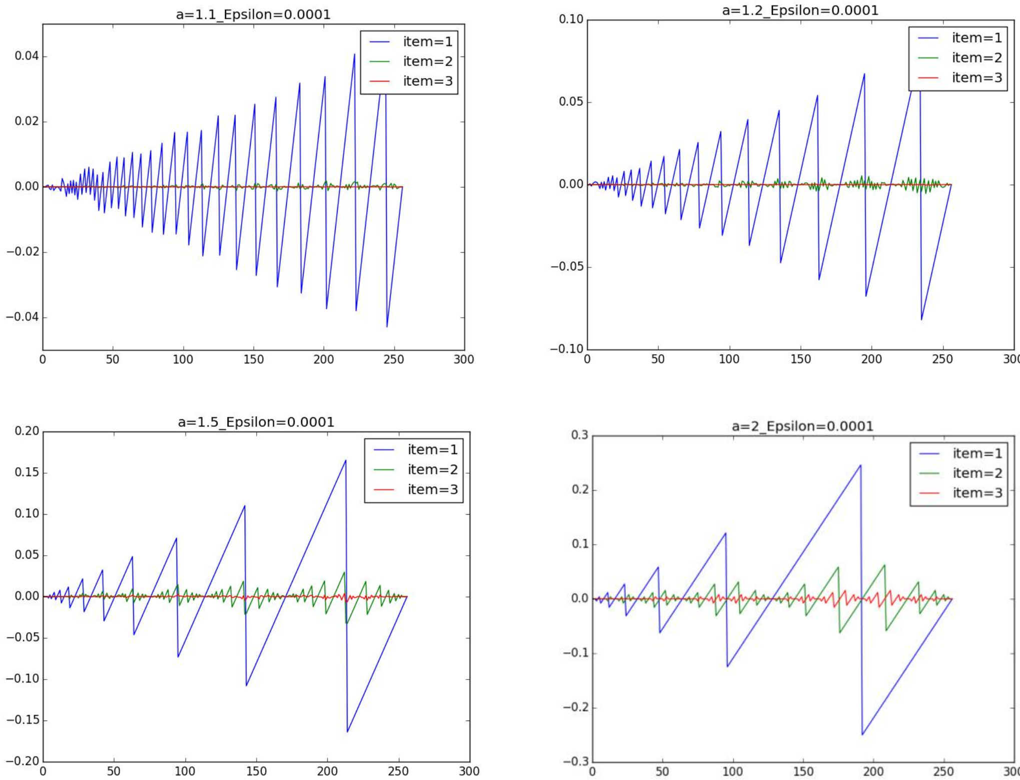

Figure 1. floating point fitting based on exponential addition

Remark 2: Each floating point can be fitted by a combination of exponential, then the operation of multiplication can be converted to addition. Given any small fitting error ε, there exists a combination that fits the original floating point with the residual smaller than ε.

Based on remark 1 and remark 2, we show that CNNs are robust to slight parameter change, and any parameter can be replaced by a combination of exponential with arbitrary small fitting error. Therefore, we can make a conjecture, given as follows:

Conjecture 1： *Given any CNN, each parameter can be replaced by a accumulation of exponential. Then the multiplication operation can be converted to exponential addition. Specially, there exists a combination of exponential that satisfying any fitting error.*

If conjecture 1 is correct, then all the multiplication operations in CNN can be replaced by integer addition. This strategy may drastically improve the efficiency of CNN, which allows CNNs deploy on cheap devices.

4. Transfer Multiplication to Exponential Addition

4.1. The principle of converting multiplication of floating point to integer addition

The basic idea of our speed up strategy is to convert floating point multiplication to integer addition. Typically, a floating point C can be approximated by accumulation of a sequence of exponential, given as the following equation.

$$C \approx \sum_{i=1}^{K} a^{x_i} \tag{2}$$

For example, given N=2, the multiplication between to float values becomes:

$$\begin{aligned} C^1 \cdot C^2 &= (a^{x_1^1} + a^{x_2^1})(a^{x_1^2} + a^{x_2^2}) \\ &= a^{x_1^1+x_1^2} + a^{x_1^1+x_2^2} + a^{x_2^1+x_1^2} + a^{x_2^1+x_2^2} \end{aligned} \tag{3}$$

In particular, when N=1, the multiplication can be directly transfer to only one addition operation:

$$C^1 \cdot C^2 = (a^{x_1^1})(a^{x_1^2}) = a^{x_1^1+x_1^2} \tag{4}$$

Actually, in the section of experiments, we'll show that N=1 with small value of a is a good trade off between computational cost and accuracy lost. Since this strategy provide a general method to replace floating point multiplication, it can be easily generalized to all convolutional neural network.

4.2. The construction of look up tables

Given an input floating point, the purpose of look up table is to provide fast index of its related exponential. For example, when we dealing with a 3×3 template, the target is converted to the addition of 9 exponentials. We handle this problem by substituting the 9 exponentials with floating points. The fastest method is to acquire the 9 floating points from a look up table. We denote this look up table as exponential-real, which provides floating points given any exponential. Consequently, when we accumulate the 9 floating points, the result should be convert to exponential to the next layer. Hence the second look up table, named as real-exponential, should be constructed to give fast index of exponential combination given any floating point.

Algorithm 1. The greedy strategy for constructing real-exponential look up table

Input: The number of items K; the base of exponential a; ε ; the quantized factor M.
Output: The real-exponential look up table L_2.

Generate the floating point sequence a^0, a^1, a^2, ..., a^N, where.N=$\log_a \varepsilon$.
Construct quantized intervals $[\frac{m}{M}, \frac{m+1}{M}]$, m={0,1,...,M-1}according to quantized factor M.
foreach quantized interval $[\frac{m}{M}, \frac{m+1}{M}]$ {
    $r_1 = \frac{m}{M}$;
    for (i = 1: K){
        r_{i+1}=MaxNum;
        for (j = 0: N) {
            tmp=abs(r_i - a^j);
            if ($tmp < r_{i+1}$){
                r_{i+1}=tmp; $C[i]$=j;
            }//endif
        }//endfor
        L[m][i] = $C[i]$; // Update look up table
    }//endfor
}//endfor

Obviously, the length of real-exponential look up table is equal to the range of N. Then each floating point in the look up table is directly derived from the item. Namely, $L_1[i]=a^i$, i={-1, -2, ..., -N}. As for the real-exponential look up table, we use a greedy scheme to establish the mapping between a floating point and it corresponding exponential. As shown in Algorithm 1, for each floating point falls in the quantized interval $[\frac{m}{M}, \frac{m+1}{M}]$, we aims at giving the related exponential from the look up table. Since slight changes of the value of parameter in CNN doesn' t influence the performance of the network, we then provide the same integer combination for each floating point in $[\frac{m}{M}, \frac{m+1}{M}]$. Suppose the combination has K items, then the the greedy scheme is performed as the process which is mentioned in Theorem 1. Details of the algorithm for constructing real-exponential look up table is given in Algorithm 1. Since the two look up tables are all built off-line, the constructing process doesn't influence the efficiency of the whole algorithm.

4.3. The pipeline of exponential addition in CNNs

In CNN, the multiplication operation exists in convolutional layer and fully connect layer. Since we use different strategies in the two layer, they are discussed separately. As for convolution operation layer, the kernels are expressed as the form of exponential. As a consequent, the pixel value of input image/feature map should be converted to an integer, which is acquired from the real-exponential look up table. For each pixel $I(x,y)$ in the input image/feature map, the convolution operation is conducted according to the principle that mentioned in subsection 4.1. Consequently, the result always contains the accumulation of several items. Then the exponential-real look up table is used to covert each item to a floating point. After finishing the summation in a template, the result should be converted to a exponential again via the real-exponential look up table. The details are given in Algorithm 2.

Algorithm 2. The pipeline of convolution without multiplication in a quantized network4.4 Time complexity analysis

Input: The number of items K; the base of exponential a; ε ; the quantized factor M; The look up tables L_1 and L_2; The quantized convolutional kernel C; The input feature map/image I; The target neural p.
Output: The output signal of p.

if (p in convolutional layers){//perform convolution
    Select a template T, which is centered on p's coordinate, in p's former layer.
    Perform exponential addition to replace multiplication convolution operation K on T. Denote the results of exponential sequence as n_i, where i={1, 2, ..., scale(C)}
    for(i=1: scale(C)){
        if (n_i>N) continue; // The related floating point is 0.
        else{
            search related floating point of n_i in the exponential-real look up table. Accumulate the result.}//else
    }//for
else{ //p in fully connect layers
    Construct a table T3 to record the frequence of each exponential.
    for(j=1: the number of neurals in former layer){
        Perform exponential addition: tmp=Np+Nq_j //q_j denotes the jth neural in the former layer.
        Perform frequence accumulation: T3[Np+Nq_j]++
    }//for
    Locate all non-zero elements of T3 in the exponential-real look up table. Accumulate the result.
    Convert the floating point to the related exponential via the real-exponential look up table. Output the result.
}//else

As for CNN with fully connected layer, each neural connects with every neural in the former layer. Namely, if the former layer has M neurals, then there would be M multiplication operations to produce an output. In order to avoid visiting the look up table frequently, we replace the repetitive visiting with a histogram searching scheme. For example, in the VGG network, the number of neurals in the penultimate layer and the output layer are respectively as 4,096 and 1,000. In other words, each output neural connects with 4,096 neurals in the penultimate layer. It is worth noting that in our experiments, the range of N is typically smaller than 100, therefore there exists many repeated numbers. This phenomenon gives us a chance to drastically reduce the frequency of visiting look up tables. In view of this characteristic, we construct a table to store the frequency of each exponential been hit.

5. Experiments

5.1. Description of the experiments

In this section, we aim at evaluating the performance of the proposed accelerate scheme on mainstream models, including AlexNet[13], VGG[23], Residual Net[11] and Faster-RCNN[20]. The main task is to compare the accuracy loss between our quantized model and its original one. The detailed parameter settings in the implementation are given as follows. As for the task of image classification, we test all kinds of combinations, namely the base a, the minimal loss ε, the number of items K. Firstly, since a ranges from [1, 2], then a is respectively set as 1.1, 1.2, 1.5, 2.0. As for the minimal loss ε, it directly relates to the length of real-exponential look up table. We then choose $\varepsilon=\{10^{-4}, 10^{-3}, 10^{-2}, 0.02, 0.04\}$ to comprehensively evaluate the influence of ε. In addition, the number of items determine the length of exponential-real look up table in quantized models. Therefore, this parameter is a key factor in the proposed scheme, which makes a trade off between speed and accuracy. According to the experiment results shown in the section of motivation, the combination of three items has highest fitting accuracy compare its related floating point. Actually, our experiments show that the quantized models have almost identical performances with two items. As a consequent, K is respectively been set as 1, 2. As for the task of object detection, ε, a, and the number of items are respectively set as $\varepsilon=\{10^{-4}, 10^{-3}, 10^{-2}, 0.05\}$, a=\{1.1, 1.2, 1.5, 2.0\}, and item=\{1, 2\}.

Moreover, in order to evaluate the efficiency of the proposed method, some metrics have been employed such as top1 and top5 in the task of image classification, and mAP in the task of object detection. The results acquired from original convolutional networks are regarded as the baseline.

5.2. Image Classification

The first experiment is to evaluate the performance of our strategy on the image classification task. Since ε is a key parameter of the proposed scheme, the experiment then focuses on the accuracy loss with scaling the value of ε.

Firstly, we try to test the performance of quantized AlexNet, VGG, and ResNet with small value of ε. For example, $\varepsilon=10^{-4}$ means the minimal interval in the quantized model is $[0, 10^{-4}]$. As mentioned before, the length of exponential-real look up table is $N=-\log_a \varepsilon$. Table 2 shows the results of three quantized models with $\varepsilon=10^{-4}$. One can easily see that when we choose small value of a, for example a=1.1 or a=1.2, the number of item doesn't influence the classification accuracy. In particular, some quantized models even outperform the original model. For example, when a=1.1, item=2, and a=1.2, item=2, the top1 and top5 in quantized VGG, and top1 in quantized ResNet50 are all better than the original model. This phenomenon reveals the fact that small value of a leads to small quantized intervals near to 1, which needs less item to fit floating points near to 1. Therefore, the item value is not a crucial factor, since the quantized models only have slight accuracy loss in most cases.

On the other hand, when a has larger values, the length of intervals varies greatly. As is shown in Fig.1, when a=2 with item =1, the values near 1 would has larger residual than the result with smaller a. As a consequent, the quantized model performs worse than their original models. For example, when $\varepsilon=10^{-4}$ with a=2.0 and item=1, the accuracy are drastically dropped.

Table 2. The performance of quantized models with $\varepsilon=10^{-4}$ for Image Classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	70.956	89.9161	75.102	**92.2002**	**56.8**	79.946
a=1.1, item=1	70.636	89.8321	74.348	91.8062	56.564	79.814
a=1.1, item=2	**70.9561**	**89.9261**	**75.1582**	92.1762	56.726	**79.98**
a=1.2, item=1	70.1619	89.4501	72.0562	90.2943	55.912	79.244
a=1.2, item=2	**70.9579**	**89.9242**	**75.1743**	92.1841	56.764	79.946
a=1.5, item=1	65.4239	86.1862	49.032	72.3322	51.418	75.54
a=1.5, item=2	70.9039	89.8821	74.8763	92.0701	56.668	79.936
a=2.0, item=1	45.07	68.8479	7.25805	16.528	41.656	65.848
a=2.0, item=2	70.396	89.5382	73.7122	91.2761	56.386	79.63

Table 3. The performance of quantized models with $\varepsilon=10^{-3}$ for Image Classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	70.956	89.9161	75.102	**92.2002**	**56.8**	79.946
a=1.1, item=1	70.6899	89.8261	74.48	91.7874	56.602	79.796
a=1.1, item=2	**70.9599**	89.9142	**75.1363**	92.1691	56.782	**79.962**
a=1.2, item=1	70.1619	89.4541	71.8868	90.1663	55.9	79.24
a=1.2, item=2	**70.984**	**89.9222**	75.0083	92.1787	56.738	79.936
a=1.5, item=1	65.4299	86.1902	48.7552	72.2563	51.466	75.542
a=1.5, item=2	70.9059	89.8741	74.8339	92.0157	56.706	79.964
a=2.0, item=1	45.072	68.8479	7.30079	16.4912	41.614	65.85
a=2.0, item=2	70.398	89.5382	73.6179	91.2477	56.434	79.604

Table 4. The performance of quantized models with $\varepsilon=10^{-2}$ for Image Classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	**70.956**	**89.9161**	**75.102**	**92.2002**	**56.8**	**79.946**
a=1.1, item=1	70.6339	89.7461	74.3082	91.7643	56.652	79.818
a=1.1, item=2	70.8139	89.8142	74.7803	91.9701	56.652	79.818
a=1.2, item=1	69.94	89.3961	71.8543	90.0962	55.838	79.3
a=1.2, item=2	70.666	89.7862	74.7942	91.9942	56.69	79.85
a=1.5, item=1	65.244	86.0842	48.614	72.0002	51.478	75.512
a=1.5, item=2	70.712	89.8282	74.3203	91.7801	56.61	79.774
a=2.0, item=1	41.756	65.6879	7.03446	15.954	41.492	65.74
a=2.0, item=2	68.9639	88.6362	71.5842	90.1043	56.014	79.346

The same conclusions can be drawn in Table 3 and Table 4, where the combination of small a with larger number of item has better performance. Specially, when $\varepsilon=10^{-3}$, VGG with a=1.1 or 1.2, item=2, ResNet and AlexNet with a=1.1, item=2 are all better than original models. Note that item=1 means the each floating point multiplication can be directly converted on only one integer addition. Although Fig.1 denotes that the fitting error with item=1 is large, the results from Table 2 to Table 4 show that the performances of quantized models only have slight accuracy loss with small values of a. There are two reasons can explain this phenomenon. First, deeper neural networks have larger number of parameters, whose performances are more robust to slight parameter changes. Second, most parameters in DNNs are distributed around 0. Therefore, even the parameters near 1 are replaced with large residual, the results are still promising. From the viewpoint of quantized scheme, these two factors explain the reasons why the proposed quantized strategy can accelerate the feed forward operations with promising performance.

Table 5. The performance of quantized models with ε=0.02 for Image Classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	70.956	89.9161	75.102	92.2002	56.8	79.946
a=1.1, item=1	69.734	89.21	73.61	91.2562	56.284	79.562
a=1.1, item=2	69.988	89.382	73.824	91.3722	56.536	79.684
a=1.2, item=1	68.8401	88.5761	71.116	89.6542	55.542	79.022
a=1.2, item=2	69.872	89.1721	73.1239	90.9442	56.236	79.522
a=1.5, item=1	62.61	83.8639	43.69	67.298	50.796	74.818
a=1.5, item=2	68.268	88.252	68.1699	87.9342	55.854	79.254
a=2.0, item=1	38.34	61.676	4.80803	11.764	50.508	64.84
a=2.0, item=2	62.028	84.098	61.784	83.3162	54.982	78.462

Table 6. The performance of quantized models with ε=0.04 for Image Classification

	VGG16		ResNet50		Alexnet	
	top1	top5	top1	top5	top1	top5
Original	70.956	89.9161	75.102	92.2002	56.8	79.946
a=1.1, item=1	58.542	81.3379	61.6359	83.8001	54.934	78.402
a=1.1, item=2	58.466	81.1999	62.306	82.4041	54.868	78.484
a=1.2, item=1	56.874	79.936	54.79	78.4601	53.834	77.5
a=1.2, item=2	57.962	80.9039	56.678	80.7361	54.362	78.022
a=1.5, item=1	12.856	27.392	8.174	18.486	44.162	69.056
a=1.5, item=2	18.42	36.498	15.664	31.856	50.296	74.796
a=2.0, item=1	6.88	16.85	0.866	2.60602	34	57.832
a=2.0, item=2	4.178	10.954	4.46203	10.544	47.494	72.388

Secondly, according to the definition of look up table length N=-log$_a$ ε, once we increase the value of ε, the length of exponential-real look up table will be shorten. However, the fitting error between the combination of exponential and its original floating point would be increased. Table 5 and table 6 show the results of ε=0.02 and ε=0.04 . One can see that when ε=0.02 with a=1.1or a=1.2, the quantized models still have good performances. Nevertheless, when a>1.5, the performances are drastically dropped. Table 6 also has the same property. In other words, smaller ε means most floating points are hard to be replaced by a exponential sequence with small residual. Therefore, the parameters in each quantized model are greatly different with the original one. As a consequent, the performances may dropped drastically.

Based on the experiment results mentioned above, we can draw a conclusion that ε, a, *item* are all key parameters in the proposed scheme, which makes trade off between speed and accuracy. Smaller value of ε means larger chance to select a exponential sequence with slight residual to replace a floating point. On the other hand, larger value of a means shorten look up table and faster speed, while the number of exponential combinations would be reduced, which may influence the performance. Last but not the least, the number of items directly relate to the scale of fitting error. More items mean less residuals, resulting in higher computational complexity during the process of floating point fitting and look up table searching.

5.3. Object Detection

In the second experiment, we aim at evaluating the proposed quantized scheme on the task of object detection. To this end, the Faster R-CNN, one of the most widely used detection framework, has been selected. Since our target is to evaluate the total performance of our strategy, we only concern on the Mean Average Precision (mAP) of the detect results. The experiments are conducted on PASCAL VOC 2012, which is widely used in object detection task. In particular, the result of original Faster R-CNN (mAP=69.45) is set as the baseline.

Table 7 Performances of quantized Faster R-CNN model on PASCAL VOC 2012 with differnet parameters setting

Quantized model	ε=10^{-4}	ε=10^{-3}	ε=10^{-2}	ε=0.05
a=1.1, item=1	68.53	68.53	68.21	61.81
a=1.1, item=2	69.18	68.90	68.59	61.89
a=1.2, item=1	67.74	67.74	67.98	59.11
a=1.2, item=2	69.00	68.85	68.93	60.42
a=1.5, item=1	61.20	61.20	60.40	40.11
a=1.5, item=2	68.59	68.79	68.79	49.35
a=2.0, item=1	44.32	44.32	44.20	23.06
a=2.0, item=2	68.09	68.09	68.00	25.24

Table 7 shows the results of the proposed quantized method. One can see that when ε has small value, such as ε=10^{-4}, the results are comparable with the baseline, except a=2.0 with item=1. In other words, the performance of quantized Faster R-CNN doesn't drop with small value of ε. Noteworthy that this phenomenon exists in almost all combinations of a and item. In particular, when item=1 with ε≤10^{-2} and a=1.2, the mAP still very close to the baseline, which means that each floating points operations in Faster R-CNN can also be replaced by only one exponential addition without slight mAP loss. However, when we choose small value of ε, such as ε=0.05, the performance drastically dropped. This phenomenon also reveals the fact that most floating points in Faster R-CNN are also distributed around 0, which makes the fitting residuals larger than that of small value of ε. And this is the main reason why the DNN models decays with large ε.

5.4. Acceleration analysis on FPGA

In order to show the feasibility of the proposed scheme, we also evaluate the property of acceleration on FPGA. Take Xilinx Vritex-7 for example, it is a state-of-the-art FPGA system. If we optimize the use of computational resources, our integer addition scheme approaching at least 1289/d × acceleration. Where d denotes the length of look up table. For example, if we choose the VGG model for the task of image classification, with ε=10^{-4}, a=1.2 and item=1, then the length of look up table is about 52. As a consequent, the feed forward operations obtain more than 20× speed up, compare to the original VGG model. In particular, the accuracy loss of the quantized VGG model is less than 1%, which means that our scheme is suitable for deploying CNNs on low cost devices.

6. Conclusions

In this paper, we present a novel and general speed up strategy for CNNs. The basic idea is to reduce the frequency of floating point multiplication in CNNs. Therefore, each floating point in the network is replaced by the summation of a exponential series. Then the multiplication between two floating points can be converted to the addition among exponential. Specially, we give a brief description of how to convert floating point multiplication to integer addition, which is the main contribution of our work. Furthermore, we give a theorem to guarantee the range of fitting residual. In the section of experiments, the proposed scheme has been directly applied to AlexNet, VGG, ResNet for image classification, and Faster R-CNN for object detection. The results conducted on ImageNet and PASCAL VOC show that the proposed quantized scheme has promising performance. Namely, in most CNNs each multiplication can be directly replaced by only one addition operation with slight accuracy loss. These results mean that the proposed quantization scheme can be used in most CNNs, and the related quantized models can be easier deploy on mobile or other low-cost devices with limited power budgets, such as FPGA.

References

[1] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick." JMLR Workshop and Conference Proceedings, 2015.

[2] Y. Cheng, D. Wang, P. Zhou, et al. A Survey of Model Compression and Acceleration for Deep Neural Networks[J]. arXiv preprint arXiv:1710.09282, 2017.

[3] T. S. Cohen and M. Welling, "Group equivariant convolutional networks," arXiv preprint arXiv:1602.07576, 2016.

[4] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," CoRR, vol. abs/1602.02830, 2016.

[5] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 3123–3131.

[6] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1269–1277.

[7] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," CoRR, vol. abs/1412.6115, 2014.

[8] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ser. ICML'15, 2015, pp. 1737–1746.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," International Conference on Learning Representations (ICLR), 2016.

[10] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in Proceedings of the 28th International Conference on Neural Information Processing Systems, ser. NIPS'15, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.

[12] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in Proceedings of the British Machine Vision Conference. BMVA Press, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in NIPS, 2012.

[14] V. Lebedev and V. S. Lempitsky, "Fast convnets using group-wise brain damage," in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016, pp. 2554–2564.

[15] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cpdecomposition," CoRR, vol. abs/1412.6553, 2014.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," CoRR, vol. abs/1608.08710, 2016.

[17] H. Li, W. Ouyang, and X. Wang, "Multi-bias non-linear activation in deep neural networks," arXiv preprint arXiv:1604.00676, 2016.

[18] P. Merolla, R. Appuswamy, J. V. Arthur, S. K. Esser, and D. S. Modha, "Deep neural networks are robust to weight binarization and other nonlinear distortions," CoRR, vol. abs/1606.01981, 2016.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in ECCV, 2016.

[20] S. Ren, K. He, R. Girshick , et al. Faster R-CNN: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

[21] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013, 2013, pp. 2754–2761.

[22] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," arXiv preprint arXiv:1603.05201, 2016.

[23] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[24] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015, 2015, pp. 31.1–31.12.

[25] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning." CoRR, vol. abs/1602.07261, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1602.html#SzegedyIV16

[26] C. Tai, T. Xiao, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," vol. abs/1511.06067, 2015.

[27] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," CoRR, vol. abs/1702.04008, 2017.

[28] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, 2011.

[29] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2074–2082.

[30] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," CoRR, vol. abs/1612.01051, 2016.

[31] Y. Wu. Q. H. Jiaxiang Wu, Cong Leng and J. Cheng, "Quantized convolutional neural networks for mobile devices," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[32] S. Zhai, Y. Cheng, and Z. M. Zhang, "Doubly convolutional neural networks," in Advances In Neural Information Processing Systems, 2016, pp. 1082–1090.

[33] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in European Conference on Computer Vision, Amsterdam, the Netherlands, October 2016, pp. 662–677.