Advances in Intelligent Systems Research, volume 133
2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE2016)

Combining FFT and Spectral-Pooling for Efficient Convolution Neural Network Model

Zelong Wang[1,*], Qiang Lan[1,2], Dafei Huang[1,2] and Mei Wen[1,2]
[1]Department of Compute, National University of Technology Defense, Changsha 410000, China
[2]National Key Laboratory of Parallel and Distributed Processing, National University of Defense Technology, Changsha 410000, China
[*]Corresponding author

***Abstract*—Convolution operation is the most important and time consuming step in a convolution neural network model. In this work, we analyze the computing complexity of direct convolution and fast-Fourier-transform-based (FFT-based) convolution. We creatively propose *CS-unit*, which is equivalent to a combination of a convolutional layer and a pooling layer but more effective. Theoretical computing complexity of and some other similar operation is demonstrated, revealing an advantage on computation of *CS-unit*. Also, practical experiments are also performed and the result shows that *CS-unit* holds a real superiority on run time.**

Keywords-computing complexity; FFT-based convolution; CS-unit

I. Introduction

Convolution neural networks (CNNs)[11] are widely used, for their considerable performance in visual applications[6][5][9][15]. However, while delivering the impressive accuracy gain in computer vision and machine learning problems, the application of CNN is largely limited by high storage consumption and computational complexity, especially on embedded facilities.

There is some recent work aiming at reducing the complexity of CNN models. Denton et al.[4] exploited the redundancy of CNN models with linear compression techniques, resulting in significant speedups for the evaluation of trained large scale networks, with minimal compromise to performance. They reported experiments on Imagenet models[9] and showed empirical speedups on convolutional layers by a factor of $2-3\times$ and a reduction of parameters in fully connected layers by a factor of $5-10\times$. Jaderberg et al.[8] illustrated a group of tensor decomposition schemes, with the speedup going up to $4.5\times$. Given a layer, Lebedev et al.[10] computed a low-rank CP-decomposition on 4-D convolution kernel tensors into a sum of rank-one tensors. Then the original convolutional layer would be replaced with a sequence of four convolutional layers with small kernels. Their approach obtained a $8.5\times$ CPU speedup of whole network for the 36-class character classification CNN. Based on Jaderberg et al.[8], Tai et al.[16] further developed the tensor decomposition idea, proposing a new method for training low-rank constrained CNNs from scratch and a new algorithm for computing the low-rank tensor decomposition. They achieve a considerable speedup ($1.5\times$) on NIN[12] model.

Contributions above have a similar idea, that is utilizing decomposed models with less parameters to approximate original convolutional kernels. Except for decomposition schemes, Niecolas et al.[17] used FFT implementation of convolution on the Torch7[3] frame work, resulting a considerable speeding up. Micheal et al.[13] presented the back-propagation of FFT based convolution, reducing the computation complexity thus making CNN training much faster.

Pooling is a layer in a CNN model for reducing dimension of features, so as to make parameters capacity and computation not too large. A number of pooling schemes are proposed, such as max pooling[2], average pooling[11], fraction max pooling[7],etc. Rippel et al.[14] proposed *Spectral-Pooling*, which performed dimensionality reduction by truncating the representation in the frequency domain. They showed it preserved significant more information for the same number as other pooling schemes. The output map of *Spectral-Pooling* can also specify to be any arbitrary dimension. It achieves a competitive classification rate on cifar-10 and cifar-100.

In this work, we give some details about the FFT-method based convolution operation together with computing complexity analysis conducted by comparing with the direct convolution. Also, *Spectral-Pooling* will be further exploited. Unlike existed scheme that conducting *Spectral-Pooling* after the FFT based convolution, we combine FFT based convolution and *Spectral-Pooling* together and propose a *CS-unit*, which reduces a FFT and a IFFT operation. Along with detailed illustration, comparison of computational complexity is also conducted among different kinds of schemes to highlight the efficiency of our *CS-unit*.

In order to support the analytical deduction, practical experiments are also performed through comparing the actual run time of different schemes. We report the running time of FFT-method convolution and direct convolution in a single layer to illustrate that FFT-method convolution indeed has a advantage on computing complexity in practice. Also, running time of some schemes of convolution operation and pooling operation is reported. The results all reveal that *CS-unit* has the least running time among the several schemes.

II. THEORY

A. Convolution Theorem

Convolution operation is the most important and time-consuming step in a CNN model. As a common technique in signal processing, Fourier transformation on the result of convolution operation can be converted to element-wise multiplication of Fourier transformation. More clearly, we'll illustrate this procedure according to convolution theorem.

For convenience, here we only consider the 1-D operation, since 2-D (matrix) operation has a similar formation and is easy to derive.

Given with two 1D vectors: $v = (v_0, v_1, \cdots, v_{N-1})$, $w = (w_0, w_1, \cdots, w_{n-1})$, we denote the convolution operation between v and w by:

$$c_n = v * w = \sum_{k=0}^{N-1} v_k w_{n-k}, \qquad n = 0, \cdots, N-1 \tag{1}$$

Here we define $w_{-i} = w_{N-i}$ where $i = 0, \cdots, N-1$, since circular convolution is adopted here.

Then we represent the Fourier transformation on the convolution output c (the formula below is generic thus can be applied to any other 1D vectors, e.g., v and w.) using:

$$\mathcal{F}(c)_u = \sum_{n=0}^{N-1} c_n e^{-i\frac{2\pi n}{N}u}, \qquad u = 0, \cdots, N-1 \tag{2}$$

where $e^{i\theta} = \cos\theta + i\sin\theta$.

Based on the notation above, the convolution theorem can be interpreted as follows.

THEOREM 1 (CONVOLUTION THEOREM)

$$\mathcal{F}(c) = \mathcal{F}(v) \odot \mathcal{F}(w).$$

where $\odot$ denote the element-wise multiplication.

The 2-D form of the convolution theorem is the same as the theorem above expect for the different dimension of the variables. Therefore, according to the theorem, the convolution result of two matrices, X and W, can be calculated in another way:

$$X * W = \mathcal{F}^{-1}(\mathcal{F}(X) \odot \mathcal{F}(W)) \tag{3}$$

B. Computing Complexity of FFT

Former researches had explained a little about the computing complexity of two convolution schemes (direct and FFT-based). In the following discussion, we explain this idea in detail.

For a single image of size $n \times n$ with a kernel of size $k \times k$, direct convolution holds a complexity of n^2k^2, since the kernel flips for n^2 times and there are k^2 operations for

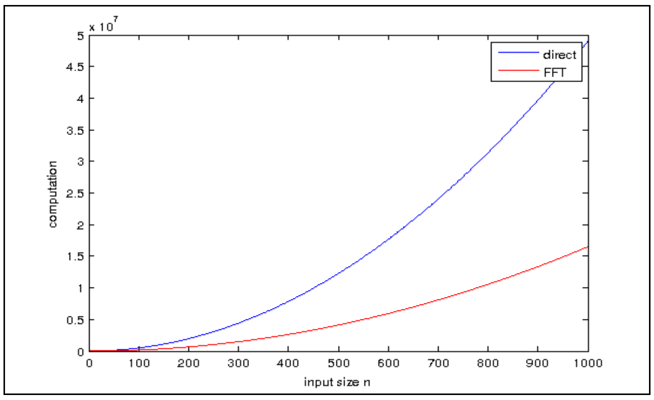

FIGURE I. ILLUSTRATION OF THE COMPUTATION OF DIRECT CONVOLUTION (THE BLUE LINE) AND FFT-BASED CONVOLUTION (THE RED LINE), WHERE KERNEL SIZE IS 7×7

each time (Here, we consider the output size is the same as the input size). According to (3), there is $\mathcal{O}(n^4)$ complexity for DFT-based convolution

In practical, we can decrease the DFT's computing complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2 \log n^2)$, by using a new implementation called fast Fourier transform (FFT). A number of FFT implementations have been proposed and there are also many languages (like MATLAB, Numpy in Python etc.) having packages to implement it.

As for FFT-based convolution, the whole computing complexity is composed by two aspects. On one hand, each FFT operation has a complexity of $\mathcal{O}(n^2 \log n^2)$ while there are three times for doing that. The first one is FFT operation on the image, the second one is that on the kernel and the third one is that on the result of multiplication. On the other hand, the multiplication operation has a complexity of $4n^2$. Therefore the whole computing complexity of FFT is $6Cn^2 \log n + 4n^2$, where C denotes a constant number. A comparison of complexity between the two schemes is demonstrated in Figure I.

According to the analysis above, we see that computing complexity of direct convolution depends on n and k, while that of FFT-based convolution only depends on n. Therefore, when given an image, the bigger k is, the more computing complexity FFT-based convolution has. This is meaningful since a big kernel can observe more information in a model. Also, a practical experiment is done in section IV

III. Fast Spectral Pooling

Spectral-Pooling proposed by Rippel et al.[14] makes CNN models get more powerful ability on classification tasks. However, when using this pooling scheme with direct convolution, a large amount of computing consumption is unavoidable. Considering the input of size $n \times n$ and kernel of size $k \times k$ as before, according to the analysis in section II.B, direct convolution holds a complexity of n^2k^2. Based on

Algorithm I: This is a description of *CS-unit*.

```
Algorithm 1 CS-unit
 Input: The image $\mathbf{x} \in \mathbb{R}^{n \times n}$, kernel $W \in \mathbb{R}^{k \times k}$
 Output: The output $\mathbf{y} \in \mathbb{R}^{m \times m}$
 1: $\mathbf{fx} \leftarrow \mathcal{F}(\mathbf{x})$
 2: $\mathbf{fw} \leftarrow \mathcal{F}(W)$
 3: $\mathbf{Prod} \leftarrow \mathbf{fx} \odot \mathbf{fw}$
 4: $\hat{\mathbf{y}} \leftarrow CropAndAdjust(\mathbf{Prod})$
 5: $\mathbf{y} \leftarrow \mathcal{F}^{-1}(\hat{\mathbf{y}})$
```

Rippel et al.[14], the complexity of computation of the *Spectral-Pooling* operation is $\mathcal{O}(2n^2 \log n) + \mathcal{O}(2m^2 \log m)$, where m denotes the output size after pooling layer. Therefore, *Spectral-Pooling* with direct convolution holds a complexity of $n^2k^2 + \mathcal{O}(2n^2 \log n) + \mathcal{O}(2m^2 \log m)$, which can also be represented as $n^2k^2 + 2C_1n^2 \log n + 2C_2m^2 \log m$, where C_1, C_2 are constant numbers.

Rippel et al.[14] mentioned a little that *Spectral-Pooling* can be used in convolution neural networks that employ FFTs for computation. However we did not find much detailed information. According to the analysis above, the complexity of the scheme that adopts *Spectral-Pooling* after FFT-based convolution is $6C_1n^2 \log n + 4n^2 + 2C_1n^2 \log n + 2C_2m^2 \log m$.

In this work, we propose a algorithm to combine a convolution operation and a pooling operation as a whole unit, making further reduction in computing complexity. We name the unit as ***CS-unit***.

Now some details of *CS-unit* is following and you can see the complete algorithm in Algorithm I.

First, we do the FFT on both image and kernel, and then element-wise multiplication will be executed. After that, some steps for *Spectral-Pooling* are executed instead of IFFT. At last we do the IFFT on the output of the *Spectral-Pooling*, and these steps are corresponding to the convolution layer with pooling layer.

The detailed steps are illustrated in Algorithm I. The *CropAndAdjust* is corresponding to *CropSpectrum* and *TreatCornerCase* referred in the contribution of Rippel et al [14]. These operations are just for cropping dimensions of output and making it meet the conjugate symmetry constrains.

Based on the Algorithm 1, complexity of *CS-unit* can be easily figured out, that is $4C_1n^2 \log n + 4n^2 + 2C_2m^2 \log m$. We compare the complexity of a group of schemes in TABLE I.

TABLE I.

Schemes	*Complexity of computation*
Direct conv + SpecPool	$n^2k^2 + 2C_1n^2 \log n + 2C_2m^2 \log m$
FFT+SpecPool	$8C_1n^2 \log n + 4n^2 + 2C_2m^2 \log m$
CS-unit	$4C_1n^2 \log n + 4n^2 + 2C_2m^2 \log m$

IV. Experiment

A. *Direct-Conv and FFT-Conv*

We report the running time of direct method convolution and FFT-based convolution, and then we illustrate the timing consumption graph to give a clear vision. We do this experiment on the Tensorflow[1], which is famous for a powerful deep-learning framework of Google.

The result illustrated in the Figure II is corresponding to the analysis in section II.B that the computation requirement will increase when kernel size becomes large.

And according to the complexity analysis of FFT-based convolution in section II.B, since the input size is fixed, the running time of FFT-based convolution remains stable.

B. *FFT Convolution with Spectral-Pooling and CS-unit*

In this part, we demonstrate the running time of three different convolution and pooling schemes (direct convolution with max-pooling, FFT convolution with *Spectral-Pooling* and *CS-unit*). The result in the Figure III shows that the running time increases with input size becoming large. It reveals that *CS-unit* is the most efficient operation among the three schemes. Here, input channel is 3 and output channel is 10 with pooling size is 64×64.

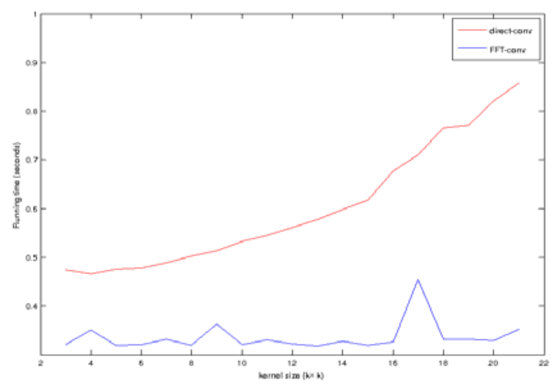

FIGURE II. EXPERIMENT RESULT OF THE RUNNING TIME OF DIRECT CONVOLUTION SCHEME AND THE FFT METHOD BASED CONVOLUTION SCHEME, WHERE INPUT SIZE IS $512 \times 512 \times 3$, OUTPUT CHANNEL = 32. WE CAN OBVIOUSLY SEE THAT TIME USAGE IS GOING UP WITH THE INCREASE OF THE KERNEL SIZE, WHILE THAT OF THE FFT METHOD BASED CONVOLUTION SUBSTANTIALLY REMAINS STABLE.

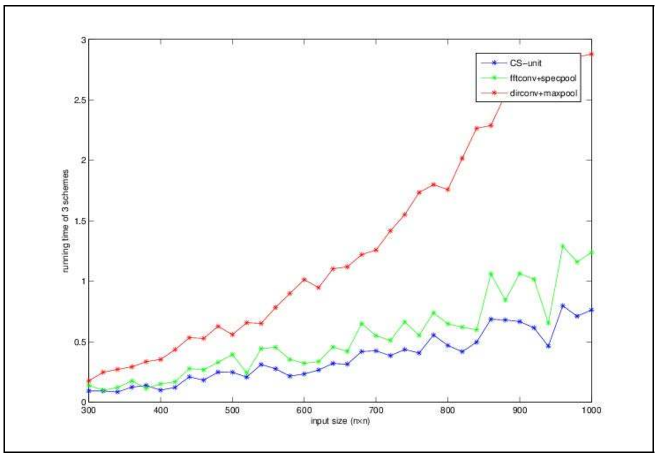

FIGURE III. RUNNING TIME OF THREE DIFFERENT SCHEMES OF CONVOLUTION OPERATION AND POOLING OPERATION IN A SINGLE LAYER. HERE, THE INPUT CHANNEL IS 3. OUTPUT CHANNEL IS 10 WITH POOLING SIZE IS 64 × 64. OBVIOUSLY CS-UNIT HAS THE LEAST RUNNING TIME.

V. Discussion

In this work, we take convolution neural network from spatial domain to frequency domain by using fast Fourier Transform based convolution and *Spectral-Pooling*. The reason for doing this work has two aspects. Firstly, FFT-based convolution has a great advantage on complexity of computation thus it can achieve a significant speedup. Secondly, as a pooling operation on frequency domain, *Spectral-Pooling* has an advantage in accuracy on classification task. We combine the two methods, making it equivalent to two operation (FFT-based and *Spectral-Pooling*) and getting a considerable speedup.

There are also some limitations in our work. For example, *CS-unit* is only available on a pre-trained CNN model. However, this problem can be solved later, since back-propagation algorithm is not difficult to implement. We will do this in follow-up work. In the future work, a main point is to implement the back-propagation algorithm of *CS-unit* and apply it on a large scale task. What is more, by using fast Fourier transform based method, great speedup is achieved. However, a large amount of storage is consumed since kernel must be padded to the same size as input to implement element-wise multiplication. Therefore, FFT method is a way that takes space for time.

Acknowledgment

The authors would like to thank Dr. Lan for his useful discussion and some creative ideas. Dafei Huang is a genius doctor in Nation University of Defense Technology. He gave many suggestions when writing this paper, thanks to him. Also, we would give many thanks to Mei Wen, who is our professor. She pointed out many details that we can make better.

The authors gratefully acknowledge supports from National Nature Science Foundation of China under NSFC No. 61502509, 61402504 and 61272145; National High Technology Research and Development Program of China (863 Program) under No. 2012AA012706.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.

[2] Y.-l. Boureau, Y. L. Cun, et al. Sparse feature learning for deep belief networks. In Advances in neural information processing systems, pages 1185–1192, 2008.

[3] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop, number EPFL-CONF-192376, 2011.

[4] 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, pages 1269–1277, 2014.

[5] R. Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[7] B. Graham. Fractional max-pooling. arXiv preprint arXiv:1412.6071, 2014.

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[10] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553, 2014.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[12] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.

[13] M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through ffts. arXiv preprint arXiv:1312.5851, 2013.

[14] O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. In Advances in Neural Information Processing Systems, pages 2449–2457, 2015.

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

[16] C. Tai, T. Xiao, X. Wang, et al. Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067, 2015.

[17] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. arXiv preprint arXiv:1412.7580, 2014.