设计研究与应用

基于卷积神经网络的 GFW 加速调度算法

宋 铁

(上海理工大学光电信息与计算机工程学院，上海 200093)

摘 要：神经网络的广泛应用使得人们更加关注神经网络的训练，更高精度的要求给神经网络的训练带来了困难，因此加速神经网络的训练成为了研究的重点。对于神经网络的训练中卷积层占据了大部分的训练时间，所以加速卷积层的训练成为了加速神经网络的关键。本文提出了 GFW 加速调度算法，GFW 算法通过对不同卷积图像的大小和卷积核的数量调用不同的卷积算法，以达到整体的最佳训练效果。实验中具体分析了 9 层卷积网络的加速训练，实验结果显示，相比于 GEMM 卷积算法，GFW 算法实现了 2.901 倍的加速，相比于 FFT 算法 GFW 算法实现了 1.467 倍的加速，相比于 Winograd 算法，GFW 算法实现了 1.318 倍的加速。

关键词：卷积神经网络；GEMM；FFT；Winograd 算法；GFW 调度算法

中图分类号：TP391; TP183 文献标识码：A **DOI**：10.3969/j.issn.1003-6970.2019.03.044

本文著录格式：宋铁. 基于卷积神经网络的 GFW 加速调度算法[J]. 软件，2019，40（3）：217-221

GFW Accelerated Scheduling Algorithm Based on Convolutional Neural Network

SONG Tie

(*School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai*, 200093, *China*)

【Abstract】: The wide application of neural networks makes people pay more attention to the training of neural networks. The requirement of higher precision brings difficulties to the training of neural networks. Therefore, the training of accelerated neural networks has become the focus of research. For the training of neural networks, the convolutional layer occupies most of the training time, so the training of the accelerated convolution network becomes the key to accelerate the neural network. In this paper, the GFW accelerated scheduling algorithm is proposed. The GFW algorithm calls different convolution algorithms on the size of different convolution images and the number of convolution kernels to achieve the overall optimal training effect. In the experiment, the acceleration training of the 9-layer convolutional network is analyzed in detail. The experimental results show that compared with the GEMM convolution algorithm, the GFW algorithm achieves 2.901 times acceleration; compared with the FFT algorithm, the GFW algorithm achieves 1.467 times acceleration; Compared to the Winograd algorithm, the GFW algorithm achieves a 1.318x acceleration.

【Key words】: Convolutional neural network; GEMM; FFT; Winograd algorithm; GFW scheduling algorithm

0 引言

自从深度学习被提出后，便迅速成为了研究的热点。深度神经网络的应用使得图像分类，语音识别和语言翻译等领域取得了很大的进步，并且在很多方面已经超过了人类的识别能力。卷积神经网络在图像和视频识别，推荐系统和自然语言处理等领域取得了较好的效果，在本文中主要研究了卷积网络训练与加速。

许多算法都被提出用以加速卷积神经网络的训练，但是每个算法都有各自的优点和缺点，并且没有一个算法可以处理所有情形的问题。在本文中，我们在 GPU 环境下测试了不同算法对卷积神经网络的加速性能。根据各个算法的特点和适合不同的卷积输入图像的大小以及卷积核数量，我们对卷积神经网络中不同的卷积层使用不用的调度策略以达

作者简介：宋铁(1992-)，男，上海理工大学光电信息与计算机工程学院硕士，主要研究方向为GPU并行加速、深度学习。

到对整个神经网络的最佳训练效果。

1 相关研究

从卷积神经网络被引入到公众中以来，有很多的研究都在研究卷积神经网络的计算与加速，但很少的研究关注卷积算法之间的不同，评估卷积算法的最佳方法是实验对比不同算法之间的性能差异。Mathieu 等人[1-2]实验得出了 FFT 算法的性能。Chetlur 等人比较了隐式 GEMM，显式 GEMM 和直接卷积算法之间的性能差异的工作[3]。Lavin 等人分析了 Winograd 算法与 GEMM 和 FFT 算法相比的优势[4]。然而，通过这些年的发展，GPU 环境中的算法实现已经变得多样化。例如，GEMM 算法有三种实现方式。虽然这些实现方式执行相同的算法，但它们的性能完全不同，因此也需要对这些卷积实现算法重新进行实验评估。

有许多研究比较了不同 DNN 框架的性能，如文献[5]和[6]。有相关的工作研究了使用 GPU 在 CUDA 平台上并行训练神经网络。Pendlebury 等人[7]提出了在 NVIDIA CUDA 平台上使用神经网络的并行加速程序，相比于 CPU 其性能提升了 80%。Honghoon Jang 等人[8]在多核的 CPU 和 GPU 上实现了基于神经网路的文本检测系统。这相比于 CPU 上执行快 15 倍，相比于 OpenMp 的 GPU 执行快了 4 倍。相关的神经网络并行加速库被开发用于加速神经的并行训练，如 cuDNN 和 cuFFT 等。并且被广泛应用于神经网络的加速训练中。

本文的工作主要是实验分析了卷积算法 GEMM、FFT 和 Winograd，分析卷积算法对于不同维度大小的输入图像以及卷积核的数量对卷积网络的计算效率。据此本文提出了 GFW 加速算法，主要根据卷积网络中不同大小的卷积特征图和卷积核的数量调用不同的卷积算法，实现整个神经网络的最佳训练效果。

2 GFW 加速调度算法

2.1 GEMM

GEMM 算法实现卷积计算的思想是将卷积计算转换成两个矩阵的乘法，这样可以减少直接卷积计算中的乘法次数，从而实现加速神经网络的训练。将卷积计算转换成两个矩阵相乘，卷积转换成两个矩阵 A 和 B，矩阵 A 的维度是 $m\times n$，矩阵 B 的维度是 $n\times k$，矩阵 C 的维度是 $m\times k$。这样卷积计算可以表示为矩阵相乘 $C=AB+C$，可以定义为公式（1）：

$$c_{i,j}=\sum_{l=0}^{n-1}a_{i,l}b_{l,j}+c_{i,j} \tag{1}$$

可以通过几个 for 循环来实现矩阵乘法，这也实现了对卷积计算的加速执行。

2.2 FFT 实现卷积

快速傅里叶变换（FFT）是离散傅里叶变换（DFT）的快速实现算法。离散傅里叶变换 DFT 的作用是实现时域上的离散复数与频域上的数值进行相互的映射，这样可以在频域中对输入数据做进一步的处理，可以在频域中实现卷积计算。对于离散傅里叶变换从时域中转换到频域中可表示为公式（2）：

$$X_k=\sum_{n=0}^{N-1}x_n\cdot e^{-i2\pi kn/N} \tag{2}$$

在频域中对输入数据进行计算后通过反向傅里叶变换将计算结果转换到时域中，反向傅里叶变换 IDFT 的计算可表示为公式（3）：

$$x_n=\frac{1}{N}\sum_{k=0}^{N-1}X_k e^{i2\pi kn/N} \tag{3}$$

快速傅里叶变换 FFT 是快速求解离散傅里叶变换的一种方法，使用快速傅里叶变换实现卷积计算主要是基于时域中的卷积计算与频域中的乘法计算是等价的，并且在时域中的乘法计算与频域中的卷积计算是等价的。基于此，我们可以把在时域是的相对就比较复杂的卷积计算转换到频域中作乘法计算，减少了计算的复杂度。快速傅里叶变换实现卷积计算算法为：

Step1：分别计算 afft=fft（a），bfft=fft（b）。

Step2：计算乘法 abfft=afft×bfft。

Step3：用 IFFT 计算 abfft 的 FFT 逆变换，取得卷积计算结。

2.3 Winograd 算法

Winograd 算法实现卷积计算的原理是用非耗时的计算入加法计算比较耗时的计算操作如乘法计算，这样可以减少算法的时间复杂度，提高计算的效率。

对于一维的卷积计算，当输出为 m 时，卷积核的长度为 r 时，使用 Winograd 算法实现卷积计算所需要的乘法计算数量如公式（4）所示：

$$\mu\left(F\left(m,r\right)\right)=m+r-1 \tag{4}$$

当卷积计算为二维时，输出图像大小为 $m\times n$

时，卷积核的维度大小为 $r\times s$ 时，使用 Winograd 算法实现卷积计算所需要的乘法数量如公式（5）所示：

$$\mu\left(F\left(m\times n, r\times s\right)\right)=\mu\left(F\left(m,r\right)\right)\mu\left(F\left(n,s\right)\right)= \left(m+r-1\right)\left(n+s-1\right) \tag{5}$$

Winograd 算法是通过减少复杂的乘法计算量来减少卷积计算的时间复杂度，从而实现对卷积神经网络的加速训练的效果。

2.4　GFW 调度算法

对于神经网络的加速训练过程中，通过卷积计算和池化层后，卷积层的输入特征图维度大小不断地减小，卷积核的数量也随着卷积层的变化而变化。对于不同维度的特征图和卷积核的数量适合不同的卷积算法，同一个神经网络使用一个卷积算法计算会造成内存和计算资源的浪费，这样并不能达到神经网络整体的最佳训练性能。对此，本文提出了 GFW 加速调度算法，GFW 调度算法的思想是对不同的卷积层输入特征图维度大小卷积核的数量调度不同的卷积算法，实现每一层卷积层的最佳训练，从而提高整个神经网络的训练性能。GFW 调度算法是在 GEMM、FFT 和 Winogd 卷积算法中根据特征图维度大小和卷积核的数量来调用相应的卷积算法，实现最佳的训练性能。如图 1 给出了 GFW 加速调度算法的流程图，图中 Image_size 为输入图像的大小，Filter output 为卷积核的数量，GFW 是根据这两个参数来调度相应的卷积算法，实现神经网络的最佳训练效果。

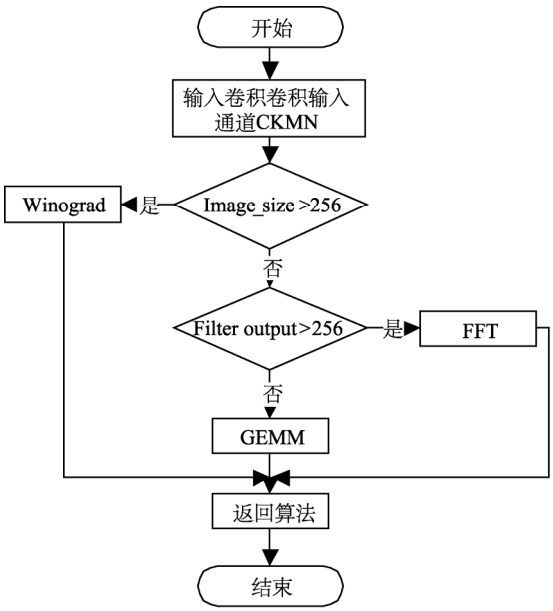

图 1　GFW 调度算法流程图

Fig.1　GFW scheduling algorithm flow chart

3　实验结果与分析

本实验完成对神经网络的加速实验，主要在同一实验平台上分别对卷积层的不同的卷积算法进行了加速实验。实验中使用的神经网络结构具有九层卷积层，分别使用 GEMM、FFT 和 Winograd 卷积算法对神经网络进行加速实验，并且与 GFW 加速调度算法进行了加速对比。实验中对神经网络结构中的每一层卷积层进行了四种卷积算法的加速实验，从实验中每一层中各个算法的执行时间可以具体分析出 GFW 加速调度算法的调度策略。最后，分析了每个卷积算法对整个神经网络的加速时间以及 GFW 加速调度算法相比其它卷积算法的加速比。

3.1　实验分析

实验中对具有九层卷积层的神经网络结构，卷积网络是由卷积层和池化层组成，输入图像经过每一层的卷积和池化层后得到的特征图都会减小，在不同卷积层的输入特征图的维度都是不同的。实验中使用的硬件平台是 NVIDIA GeForce GTX1080Ti GPU，其具有 11GB 的显存，使用了 CUDA 9.0 版本对 GPU 计算资源进行管理。实验中分别测试了神经网络每一层在四种卷积算法下的执行时间，并且分析了每个卷积算法对神经网络的加速效率。

3.2　实验结果

实验中首先分析了 GEMM、FFT 和 Winograd 卷积算法以及 DFW 加速调度算法在对九层神经网络结构的加速效果。在每一层卷积层后接着池化层，输出的特征映射图都会相比输入图片减小一半，这样神经网络结构中的不同卷积层的输入图像维度大小不同。图 2 给出了 GEMM、FFT 和 Winograd 卷积算法对神经网络中每一层的执行时间的变化以及 GFW 加速调度算法对神经网络每一层的执行效率。从图中分析可知由于每一层卷积层的输入图像维度不同，因此每一层的执行时间会随着输入特征图的减小而减小。从图中可以发现，对于 GEMM、FFT 和 Winograd 卷积算法对于神经网络卷积层的计算时间都随着输入特征图维度的减小而减小，但对于不同维度大小的输入特征图，三个卷积算法的计算时间是不同的。图中黄色曲线是 GFW 加速调度算法在加速神经网络中每一层的时间曲线，从图中可以分析出 GFW 加速调度算法始终在三条曲线的最下面，这也证明了 GFW 加速调度算法在对神经网络进行训练时，在对每一层进行计算时都会调用三个算法中计算时间最短的算法，这样可以使得神经

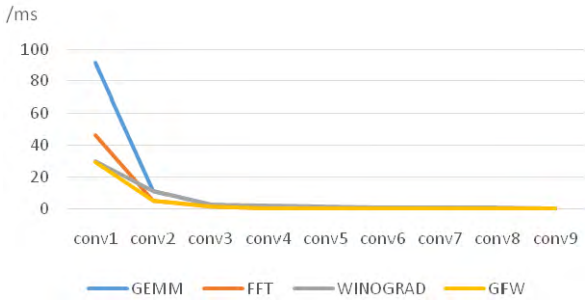

图 2 四个卷积算法对卷积网络的执行时间变化图
Fig.2 Execution time change graph of four convolutional algorithms on convolutional networks

网络的整体训练时间最小，实现神经网络的加速训练。

从图 2 分析中可知对于不同维度的输入特征图，卷积层的计算时间是不同的，并且对于相同维度输入特征图，使用不同的卷积计算算法所需要的时间也是不同的。图 3 分析了神经网络中九层卷积层使用三种不同卷积算法的执行时间,以及 GFW 加速调度算法加速卷积计算的执行时间。为了便于分析与观察，实验中将 GFW 对卷积层的计算时间进行了归一化处理，这样可以清晰的分析出在不同的卷积层中 GFW 加速调度算法相对于其它卷积算法的加速比以及 GFW 的调度策略。图中显示 Winograd 卷积算法适合于维度较大的输入特征图，而 GEMM 算法则比较适合维度较小的输入特征图，FFT 卷积算法适合于输入特征图维度居中的卷积计算。GFW 则是根据神经网络卷积层的输入特征图维度大小以及卷积核的数量来调度不同的加速算法，实现神经网络最佳的加速效果。

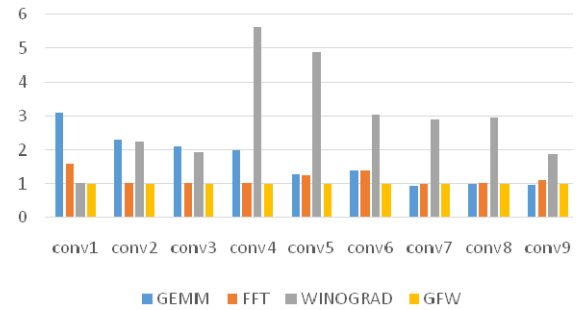

图 3 四种卷积算法对每一层卷积层的加速比
Fig.3 Acceleration ratio of four convolutional algorithms to each layer of convolutional layer

图 3 中具体分析了神经网络中每一层卷积层使用不同卷积算法的计算效率以及加速比，并分析了 GFW 加速调度算法的调度策略。GFW 加速调度算法实现每一层卷积层的调度加速从而实现了整个神经网络的加速训练，图 4 中给出了几种卷积算法对整个神经网络的加速效率。图中的三幅对比图中分别展现了几种卷积算法的计算效率，使用 FFT 实现卷积计算加速神经网络相比于使用 GEMM 实验卷积计算对神经网络的训练性能提升了 49.4%；使用 Winograd 算法实现卷积计算加速神经网络训练相比于使用 FFT 实验卷积计算对神经网络的训练性能提升了 10.2% ;本文提出的 GFW 加速调度算法对于神经网络的加速训练中，相比于目前加速效率最高的 Winograd 算法，在对神经网络的加速训练中性能提升了 24.1%。

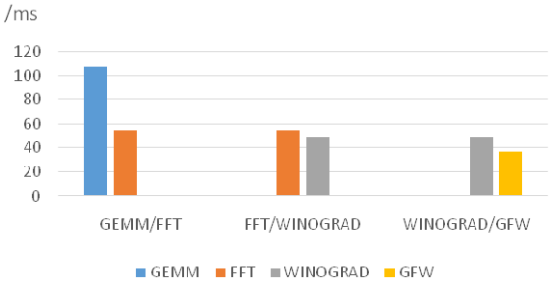

图 4 算法之间的性能提升
Fig.4 Performance improvement between algorithms

图 5 中直观的展示了本文提出的 GFW 加速调度算法相比于 GEMM、FFT 和 Winograd 算法对神经网络的加速比。在对本文具有九层卷积层的神经网络结构进行加速实验时，GFW 加速调度算法是根据卷积层中输入特征图的大小以及卷积核的数量进行调度的加速算法，因此，在对神经网络训练时加速性能较高。相比于 GEMM 卷积算法加速 GFW 调度算法取得了 2.901 倍的加速，相比于 FFT 实现卷积计算对神经网络的加速 GFW 调度算法取得了 1.467 的加速,相比于 Winograd 算法实现卷积计算，GFW 调度算法对神经网络的训练实现了 1.318 倍的加速。

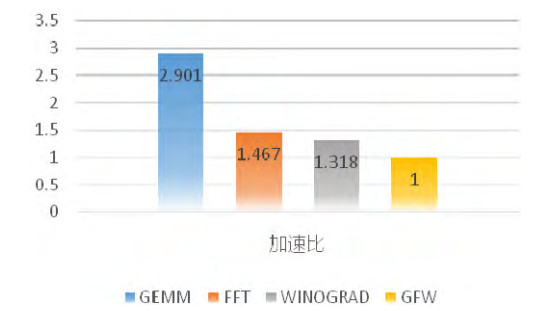

图 5 GFW 调度算法相比与其它算法的加速比
Fig.5 Acceleration ratio of GFW scheduling algorithm compared with other algorithms

4　结束语

神经网络中的卷积计算占据着神经网路的大部分训练时间，因此，加速卷积计算的效率是加速神经网络训练的关键。本文针对卷积计算提出了 GFW 加速调度算法，针对神经网络中不同卷积层的输入特征图的大小和卷积核的数量调度相应的卷积算法，实现整个神经网络的最佳训练性能。实验结果显示，GFW 加速调度算法在对神经网络的加速训练中，相比于 GEMM 卷积算法实现了 2.901 倍的加速，相比于使用 FFT 实现卷积计算对神经网络的训练，GFW 调度算法实现了 1.467 倍的加速，相比于对卷积网络计算效率较高的 Winograd 算法，GFW 调度算法在加速神经网络训练中实现了 1.318 倍的加速。

参考文献

[1] Chen T, Li M, Li Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv: 1512. 01274, 2015.

[2] Vasilache N, Johnson J, Mathieu M, et al. Fast convolutional nets with fbfft: A GPU performance evaluation[J]. arXiv preprint arXiv: 1412. 7580, 2014.

[3] Chetlur S, Woolley C, Vandermersch P, et al. cudnn: Efficient primitives for deep learning[J]. arXiv preprint arXiv: 1410. 0759, 2014.

[4] Lavin A, Gray S. Fast algorithms for convolutional neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 4013-4021.

[5] Li X, Zhang G, Huang H H, et al. Performance analysis of gpu-based convolutional neural networks[C]//2016 45th International Conference on Parallel Processing (ICPP). IEEE, 2016: 67-76.

[6] Kim H, Nam H, Jung W, et al. Performance analysis of CNN frameworks for GPUs[C]//2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE, 2017: 55-64.

[7] Pendlebury J, Xiong H, Walshe R. Artificial neural network simulation on CUDA[C]//Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on. IEEE, 2012: 228-233.

[8] Jang, Honghoon, Anjin Park, and Keechul Jung. "Neural network implementation using cuda and openmp. " Digital Image Computing: Techniques and Applications. IEEE, 2008.

[9] Xu, Rui, et al. "Accelerating CNNs Using Optimized Scheduling Strategy. " International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2018.

[10] Lu W, Yan G, Li J, et al. Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks[C]//2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2017: 553-564.

[11] Hill P, Jain A, Hill M, et al. Deftnn: Addressing bottlenecks for dnn execution on gpus via synapse vector elimination and near-compute data fission[C]//Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2017: 786-799.

[12] Song L, Wang Y, Han Y, et al. C-brain: A deep learning accelerator that tames the diversity of cnns through adaptive data-level parallelization[C]//2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2016: 1-6.

[13] Lin, Yu-Sheng, Wei-Chao Chen, and Shao-Yi Chien. "Unrolled memory inner-products: An abstract GPU operator for efficient vision-related computations. " Proceedings of the IEEE International Conference on Computer Vision. 2017.

[14] 卢冶，陈瑶，李涛，等. 面向边缘计算的嵌入式FPGA卷积神经网络构建方法[J]. 计算机研究与发展，2018，55(03): 551-562.

[15] 李景军，张宸，曹强. 面向训练阶段的神经网络性能分析[J]. 计算机科学与探索, 2018, 12(10): 1645-1657.