基于分裂基-2/(2a)FFT 算法的卷积神经网络加速性能的研究

伍家松*①②④ 达 臻①④ 魏黎明①④ SENHADJI Lotfi②③④ 舒华忠①②④
①(东南大学计算机网络和信息集成教育部重点实验室 南京 210096)
②(法国国家医学与健康研究院 U1099 雷恩 35000)
③(雷恩一大信号与图像处理实验室 雷恩 35000)
④(中法生物医学信息研究中心 南京 210096)

摘 要：卷积神经网络在语音识别和图像识别等众多领域取得了突破性进展，限制其大规模应用的很重要的一个因素就是其计算复杂度，尤其是其中空域线性卷积的计算。利用卷积定理在频域中实现空域线性卷积被认为是一种非常有效的实现方式，该文首先提出一种统一的基于时域抽取方法的分裂基-2/(2a) 1 维 FFT 快速算法，其中 a 为任意自然数，然后在 CPU 环境下对提出的 FFT 算法在一类卷积神经网络中的加速性能进行了比较研究。在 MNIST 手写数字数据库以及 Cifar-10 对象识别数据集上的实验表明：利用分裂基-2/4 FFT 算法和基-2 FFT 算法实现的卷积神经网络相比于空域直接实现的卷积神经网络，精度并不会有损失，并且分裂基-2/4 能取得最好的提速效果，在以上两个数据集上分别提速 38.56%和 72.01%。因此，在频域中实现卷积神经网络的线性卷积操作是一种十分有效的实现方式。

关键词：信号处理；深度学习；卷积神经网络；快速傅里叶变换
中图分类号： TN911.72 **文献标识码**： A **文章编号**：1009-5896(2017)02-0285-08
DOI: 10.11999/JEIT160357

Acceleration Performance Study of Convolutional Neural Network Based on Split-radix-2/(2a) FFT Algorithms

WU Jiasong①②④ DA Zhen①④ WEI Liming①④ SENHADJI Lotfi②③④ SHU Huazhong①②④
①(*The Key Laboratory of Computer Network and Information Integration* (*Southeast University*), *Ministry of Education*, *Nanjing* 210096, *China*)
②(*Institut National de la Santé et de la Recherche Médicale U* 1099, *Rennes* 35000, *France*)
③(*Laboratoire Traitement du Signal et de l'Image*, *Université de Rennes* 1, *Rennes* 35000, *France*)
④(*Centre de Recherche en Information Biomédicale Sino-français*, *Nanjing* 210096, *China*)

Abstract: Convolution Neural Networks (CNN) make breakthrough progress in many areas recently, such as speech recognition and image recognition. A limiting factor for use of CNN in large-scale application is, until recently, their computational expense, especially the calculation of linear convolution in spatial domain. Convolution theorem provides a very effective way to implement a linear convolution in spatial domain by multiplication in frequency domain. This paper proposes an unified one-dimensional FFT algorithm based on decimation-in-time split-radix-2/(2a), in which a is an arbitrary natural number. The acceleration performance of convolutional neural network is studied by using the proposed FFT algorithm on CPU environment. Experimental results on the MNIST database and Cifar-10 database show great improvement when compared to the direct linear convolution based CNN with no loss in accuracy, and the radix-2/4 FFT gets the best time savings of 38.56% and 72.01% respectively. Therefore, it is a very effective way to realize linear convolution operation in frequency domain.

Key words: Signal processing; Deep learning; Convolutional Neural Network (CNN); Fast Fourier Transform (FFT)

1 引言

深度学习是加拿大多伦多大学 Hinton 教授等人[1,2]于 2006 年提出的一种新的机器学习方式，其将无监督的逐层初始化(Layer-wise Pretraining)结构和深度神经网络(Deep Neural Networks, DNN)结构进行了有效的结合。深度学习被《麻省理工技术评论》杂志评选为 2013 年世界十大技术突破之一，吸引了学术界和工业界的广泛关注，在语音识别和图像识别等众多领域[3–5]取得了突破性进展。卷积神经网络[6–8]因为建模难度适中且性能优异，逐渐

收稿日期：2016-04-12；改回日期：2016-12-02；网络出版：2016-12-29
*通信作者：伍家松 jswu@seu.edu.cn
基金项目：国家自然科学基金(61201344, 61271312, 61401085)，高等学校博士学科点专项科研基金(20120092120036)
Foundation Items: The National Natural Science Foundation of China (61201344, 61271312, 61401085), The Special Research Fund for the Doctoral Program of Higher Education (20120092120036)

成为深度学习中众多学习结构中被广泛采纳的一种结构。限制卷积神经网络大规模应用的很重要的一个因素就是其计算复杂度，尤其是其中线性卷积的计算[9−11]。Jaderberg 等人[9]提出用低秩矩阵分解的方法来加速线性卷积的计算。Liu 等人[10]提出同时结合低秩矩阵分解和稀疏约束的方法来加速线性卷积的计算。Vasilache 等人[11]则提出在 GPU 环境下运用最简单的基-2 FFT[12]算法来加速线性卷积的实现。但是正如 Liu 等人[10]指出的虽然目前大型的卷积网络系统大部分是在 GPU 环境下实现，但是在普通的 CPU 环境中仍然有其自身的通用性的优势：基于 CPU 的系统能够方便地部署在目前通用的商品化集群系统(commodity clusters)中，而不需要任何特殊的 GPU 结点。因此研究 CPU 环境下卷积神经网络的构造也是一件十分有意义的研究工作。

虽然在卷积神经网络框架下利用快速傅里叶变换(Fast Fourier Transform, FFT)算法进行线性卷积加速的研究工作还很少[11]，但是，CPU 环境下通用 FFT 快速算法的深入研究则一直吸引着研究人员广泛的关注。目前 FFT 算法最具代表性的主要有 3 类[13]：(1)Winograd 傅里叶变换算法(Winograd Fourier Transform Algorithm, WFTA)[14]。WFTA 只能处理长度为 $N = N_1 \times N_2$ 且 N_1 和 N_2 为互质数的傅里叶变换[13]。WFTA 在 3 类算法中需要的乘法复杂度最低，加法复杂度最高，并且实现起来最为复杂。WFTA 只适用于当数据的传输代价相比与浮点数运算代价可以忽略的情况[13]，然而对于现代计算机来说，传输速度仍然比浮点数计算的速度要慢。(2)素因子算法 (Prime Factor Algorithm, PFA)[15]。PFA 同样只能处理长度为 $N = N_1 \times N_2$ 且 N_1 和 N_2 为互质数的傅里叶变换[13]。PFA 比 WFTA 需要更多的乘法，但是需要更少的加法，并且比 WFTA 更容易实现。(3)Cooley-Tukey 类型的 FFT 算法[12]。比如：Cooley 与 Tukey [12]提出了著名的基-2 FFT 算法，具有非常规则的实现结构。Duhamel 等人[16]提出了计算复杂度更低的分裂基−2/4 FFT 算法。Bouguezel 等人[17]则提出了一种统一的基于频域抽取方法的分裂基-$2/2^n$ FFT 算法，其中 n 为任意自然数。Bi 等人[18]则更进一步提出一种统一的基于频域抽取方法的分裂基-$2/(2a)$ FFT 快速算法，其中 a 为任意自然数，该算法包含了其它分裂基算法[12,16,17]，并且比素因子算法具有更低的计算复杂度。分裂基算法拥有许多其它两类算法(WFTA, PFA)所无法比拟的优势[13]，比如：(1)非常适合于处理长度为 $N = 2^n$ 的实数数据，其中 n 为任意自然数；(2)取得了计算复杂度和实现复杂度之间很好的均衡；(3)适合于同址计算，能够有效地利用规则的蝶型结构进行计算，实现程序非常简洁和紧凑；(4)具有很好地抑制量化噪声的能力；(5)适合于并行实现。需要注意的是文献[16−18]的方法都是基于频域抽取的 FFT 算法。但是在有些应用场合，比如滑动窗 FFT 算法[19−21]，为了与时域滑动过程相吻合，必须采用基于时域抽取的 FFT 算法。

本文主要研究不考虑 GPU 硬件加速的情况下，基于时域抽取的 FFT 算法在 CPU 环境下对卷积神经网络的加速性能。本文剩下的几个部分组织结构如下：第 2 节提出一种统一的基于时域抽取方法的分裂基-$2/(2a)$ 1 维 FFT 快速算法，并对其计算复杂度进行了分析；第 3 节利用经典的行列方法将提出的 1 维 FFT 算法推广到 2 维，并对得到的 2 维 FFT 算法的计算复杂度进行了分析，并且在卷积神经网络的框架下介绍了 FFT 算法的应用；第 4 节通过实验研究了基于时域抽取方法的分裂基-$2/(2a)$ FFT 算法对卷积神经网络的加速性能。

2 1 维分裂基−2/(2a)FFT 算法及其复杂度分析

1 维离散傅里叶变换(Discrete Fourier Transform, DFT)及其反变换分别定义为[22]

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \tag{1}$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) W_N^{-nk} \tag{2}$$

其中，$W_N = \mathrm{e}^{-\mathrm{j}2\pi/N}$, $0 \le n \le N-1$, $0 \le k \le N-1$, N 为序列长度。

Bi 等人[18]提出基于频域抽取的分裂基-$2/(2a)$ FFT 算法，该算法将 DFT 的系数 $X(k)$拆分为偶数项 $X(2k)$和奇数项 $X(2ak+l)$，其中 a 为任意自然数，l 为奇数变量并且$1 \le l < 2a$。偶数项 $X(2k)$的计算使用基−2 FFT 的分解方式：

$$X(2k) = \sum_{n=0}^{N/2-1} \left[x(n) + x\left(n + N/2\right) \right] W_{N/2}^{nk} \tag{3}$$

而奇数项 $X(2ak+l)$则可以表达为

$$X(2ak+l) = \sum_{n=0}^{N/(2a)-1} W_N^{nl} \left[\sum_{i=0}^{a-1} x'\left(n + iN/(2a)\right) W_{2a}^{il} \right] W_{N/(2a)}^{nk} \tag{4}$$

其中，$x'(n) = x(n) - x(n + N/2)$, $n = 0,1,\cdots,N/2-1$。这样，一个长度为 N 的 DFT 就被分解为一个 $N/2$ 长度的 DFT 和 a 个长度为 $N/(2a)$的 DFT。

下面我们在 Bi 等人[18]提出的频域抽取分裂基-$2/(2a)$ FFT 算法的基础上，提出一种新的时域抽取分裂基-$2/(2a)$ FFT 算法。

式(1)可以通过式(5)的方法计算：

$$\left.\begin{aligned} X(k) &= X_1^{N/2}(k) + \sum_{1\le l<2a,\text{且}l\text{为奇数}} W_N^{lk} X_2^{N/(2a),l,a}(k) \\ X_1^{N/2}(k) &= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} \\ X_2^{N/(2a),l,a}(k) &= \sum_{n=0}^{N/(2a)-1} x(2an+l) W_{N/(2a)}^{nk} \end{aligned}\right\} \quad (5)$$

其中 $0 \le k \le N-1$。X_2的上标分别表示：子序列长度为$N/(2a)$, a为任意自然数，$1 \le l \le 2a$，且l为奇数。

利用旋转因子W_N^k的对称性$\left(W_N^{k+N/2} = -W_N^k\right)$和周期性$\left(W_N^{k+N} = W_N^k\right)$，式(5)能够进一步转化为式(6)的矩阵形式：

$$\begin{bmatrix} X\left(k+qN/2a\right) \\ X\left(k+(q+a)N/2a\right) \end{bmatrix} = \boldsymbol{H}_2 \begin{bmatrix} X_1^{N/2}(k+qN/2a) \\ \sum_{1\le l<2a,\ \text{且}l\text{为奇数}} W_N^{l(k+qN/(2a))} X_2^{N/(2a),l,a}(k) \end{bmatrix}, \quad 0 \le q \le a-1,\ 0 \le k \le N/(2a)-1 \quad (6)$$

其中，$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$。从式(6)可以看出我们已经将一个长度为$N$的DFT分解成了 1 个长度为$N/2$ 的DFT以及a个长度为$N/(2a)$的DFT。 当$a=1$时，式(6)对应的是时域抽取基-2 FFT算法；当$a=2$时，式(6)对应的是时域抽取分裂基-2/4 FFT算法。

下面分析 1 维 DFT 正变换的计算复杂度。1 次复数乘法相当于做 4 次实数乘法和 2 次实数加法。1 次复数加法相当于做 2 次实数加法。分裂基 -2 /(2a) FFT 算法 (a = 1,2) 需要的理论上的乘法次数$\left(M_N^2,\ M_N^{2/4}\right)$和加法次数$\left(A_N^2,\ A_N^{2/4}\right)$如表 1 所示。

3　卷积神经网络与分裂基-2/(2a)FFT 算法的结合

3.1 2 维 FFT 算法应用于 CNN 中的计算复杂度分析

2 维 DFT[22]可以利用经典的行-列算法，通过 1 维 DFT 实现如式(7)所示。

$$\begin{aligned} X(k_1,k_2) &= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1,n_2) W_{N_1}^{n_1k_1} W_{N_2}^{n_2k_2} \\ &= \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x(n_1,n_2) W_{N_2}^{n_2k_2}\right) W_{N_1}^{n_1k_1} \end{aligned} \quad (7)$$

由式(7)可知，2 维 DFT 正变换可以通过先对列做 N_2个长度 N_1的 1 维 DFT，再对行做 N_1个长度 N_2的 1 维 DFT 来实现。因此 2 维 FFT 算法需要的计算复杂度为 1 维 FFT 算法的(N_1+N_2)倍。

3.2 卷积神经网络框架下的 FFT 算法的应用

目前在卷积神经网络框架下利用 FFT 算法进行线性卷积的加速，通常优先选择自己开发其中的 FFT 算法模块(比如：已经取得成功的 fbfft[11]算法)，而不是直接选择相对成熟的 FFTW 软件包[23]和 FFTE 软件包[24]。原因主要包括：(1)卷积神经网络框架下不但需要做线性卷积操作，还需要进行反向传播等操作，自己开发的 FFT 算法能够更好地与反向传播等操作相结合[11]；(2)在某些应用场合(比如：基于滑动窗的图像检测[25])，为了进一步加速线性卷积的计算需要用到滑动窗 DFT 算法[19,20]，通用的 FFTW, FFTE 软件包不容易与滑动 DFT 算法结合，此时，使用基于时域抽取的分裂基 −2 /(2a) 算法更容易与滑动 DFT 算法结合。

对于网络中给定的一层，令 P 幅输入图像集合为$\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_f, \cdots, \boldsymbol{X}_P\}$，其中 $\boldsymbol{X}_f$ 表示第 f 幅 2 维图像矩阵，大小为 $N\times N$；与该层相连的权值矩阵为 $\boldsymbol{W}_{f'f}$，大小为 $K\times K$；该层的 Q 幅输出特征图像集合为$\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \cdots, \boldsymbol{Y}_{f'}, \cdots, \boldsymbol{Y}_Q\}$，其中 $\boldsymbol{Y}_{f'}$表示第 f' 幅输出特征图像(output feature map)，大小为 $(N-K+1)\times(N-K+1)$。

对于一般的反向传播算法[11](Back Propagation, BP)，在前向传播的过程中，需要对输入矩阵做卷积，即

$$\boldsymbol{Y}_{f'} = \sum_f \boldsymbol{X}_f * \boldsymbol{W}_{f'f} \quad (8)$$

而在反向传播的过程中，需要计算整个网络的损失函数 L 关于该层输入的梯度，即

$$\frac{\partial L}{\partial \boldsymbol{X}_f} = \frac{\partial L}{\partial \boldsymbol{Y}_{f'}} * \boldsymbol{W}_{f'f}^{\mathrm{T}}, \quad (9)$$

表 1　分裂基 -2/(2a) FFT 算法计算复杂度分析(a=1,2)

FFT 算法	计算复杂度表达式	初始值
基-2	$M_N^2 = 2N\log_2 N - 6N + 8$ $A_N^2 = 3N\log_2 N - 3N + 4$	$M_4^2 = 0, M_1^2 = 0, A_4^2 = 16, A_1^2 = 4$
分裂基-2/4	$M_N^{2/4} = (4/3)N\log_2 N - 38N/9 + (2/9)(-1)^{\log_2 N} + 6$ $A_N^{2/4} = (8/3)N\log_2 N - 16N/9 - (2/9)(-1)^{\log_2 N} + 2$	$M_4^{2/4} = M_2^{2/4} = 0, A_4^{2/4} = 16, A_2^{2/4} = 4$

梯度值$\left(\partial L/\partial \boldsymbol{X}_f\right)$又会被向前一层传播，经过计算又可得到前一层的损失关于输入的梯度。最终需要求的其实是每一层损失函数关于权值矩阵的梯度，即

$$\frac{\partial L}{\boldsymbol{W}_{f'f}}=\frac{\partial L}{\partial \boldsymbol{Y}_{f'}}*\boldsymbol{X}_f \tag{10}$$

将 FFT 算法应用于以上过程可以得到以下两小节中的算法(注意：下面算法中每一层的输出隐含了激活函数的作用)。

3.2.1 前向计算算法

输入：某一层的某个输入矩阵 $\boldsymbol{X}_f$，以及相应的权值矩阵 $\boldsymbol{W}_{ff'}$。

输出：该层的输出特征值矩阵 $\boldsymbol{Y}_{f'}$。

(1) 将权值矩阵 $\boldsymbol{W}_{ff'}$“补零”到与输入矩阵 $\boldsymbol{X}_f$ 相同的尺寸，这里假设 $N=2^n$，n 为自然数，使用分裂基-$2/(2a)$ FFT 正变换(式(7))将其变换到频域中，得到频域中的权值矩阵 $\boldsymbol{FW}_{ff'}$。

(2) 将输入矩阵 $\boldsymbol{X}_f$ 同样进行分裂基-$2/(2a)$ FFT 正变换，得到频域中的输入矩阵 $\boldsymbol{FX}_f$。

(3) 计算 $\boldsymbol{FY}_{f'}=\boldsymbol{FW}_{f'f}\circ\boldsymbol{FX}_f$，“$\circ$”表示 Hadamard 乘积；同时为每一个输出特征矩阵加上该滤波器(权值矩阵)对应的偏置值的 N^2 倍。

(4) 对 $\boldsymbol{FY}_{f'}$ 做分裂基-$2/(2a)$ FFT 的反变换，并提取有效值区域$[K-1,N]\times[K-1,N]$，即可得到输出特征值矩阵 $\boldsymbol{Y}_{f'}$。

3.2.2 反向求梯度算法

输入：假设本层为第 m 层，则需要第 $m+1$ 层计算得到的损失函数关于第 $m+1$ 层输入的梯度矩阵 $\boldsymbol{\delta}_{m+1}$；第 m 层的损失函数关于输出值的梯度(即局部梯度) $\partial L/\partial \boldsymbol{Y}_{f'}$，本层预先保存的频域中的权值矩阵 $\boldsymbol{FW}_{ff'}$；以及第 $m-1$ 层的输出特征矩阵 $\boldsymbol{FY}_{f'}^{m-1}$。

输出：每一个通道的损失函数 L 关于偏置的梯度 $\boldsymbol{\delta}_b$，损失函数 L 关于本层权值矩阵(滤波器)的梯度 $\partial L/\partial \boldsymbol{W}_{f'f}$，以及损失函数关于第 $m-1$ 层输出的梯度矩阵 $\boldsymbol{\delta}_{m+1}$。

(1) 计算 $\boldsymbol{\delta}_b$：对每个滤波通道，求 $\boldsymbol{\delta}_b=\sum_{i,j}(\boldsymbol{\delta}_{m+1})_{i,j}$。

(2)计算 $\boldsymbol{\delta}_m=\boldsymbol{\delta}_{m+1}\circ\left(\partial L/\partial \boldsymbol{Y}_{f'}\right)$；

(3)将 $\boldsymbol{\delta}_m$ 补零到大小为 $N\times N$ 并旋转 180°，并对其做分裂基-$2/(2a)$ FFT 的正变换，得到 $\boldsymbol{F\delta}_m$；

(4)计算 $\partial L/\partial \boldsymbol{W}_{f'f}$：首先，计算频域中的权值梯度 $\boldsymbol{F\delta}_{\boldsymbol{W}_{f'f}}=\boldsymbol{F\delta}_m\circ\boldsymbol{FY}_{f'}^{m-1}$；然后，将 $\boldsymbol{F\delta}_{\boldsymbol{W}_{f'f}}$ 进行分裂基-$2/(2a)$ FFT 的反变换，并提取有效值区域$[N-K+1,N]\times[N-K+1,N]$，而后旋转 180°，即可得到 $\partial L/\partial \boldsymbol{W}_{f'f}$。

(5)计算 $\boldsymbol{\delta}_{m-1}$：首先计算频域中的第 $m-1$ 层损失关于第 $m-1$ 层输出的梯度：$\boldsymbol{F\delta}_{m-1}=\boldsymbol{FW}_{f'f}\circ\boldsymbol{F\delta}_m$，然后对 $\boldsymbol{F\delta}_{m-1}$ 做分裂基-$2/(2a)$ FFT 的反变换，并提取有效值区域$[1,N]\times[1,N]$，而后旋转 180°，即可得到空域中的 $\boldsymbol{\delta}_{m-1}$。

3.3 整个卷积神经网络在频域和空域中的计算复杂度近似分析

由于 CNN 的复杂性，想要精确计算整个卷积神经网络所需的加法次数和乘法次数是比较困难的。因此，这里使用所需的全部的运算次数这样一个稍微不精确的度量来衡量 CNN 的计算复杂性。众所周知，Cooley-Tukey FFT 的复杂度总体来说都为 $O(N\log_2 N)$，各种方法的差异仅在于一个常数，为方便起见，忽略这个常数。假设对于网络中的某一层来说，输入图像大小为 $N\times N$，卷积核的大小为 $K\times K$，输入图像的数量为 P，输出的特征图数量为 Q，每 S 个样例更新权值 1 次(即 batchsize=S)。在上述参数设置的情况下，空域直接线性卷积需要 $(N-K+1)^2N^2$ 次运算，而运用 FFT 算法计算线性卷积则大约只需要 $O\left(N^2\log_2 N^2\right)=2CN^2\log_2 N\approx 2N^2\log_2 N$ 次运算，这里忽略常数 C 是合理的：因为分别在空域和在频域中做线性卷积时，两种方法的计算复杂度的数量级已经不同，忽略常数 C 并不会对结果有太大的影响。

在空域中某一层迭代一次所需的运算次数为(包括前向传播和后向反馈的过程)[11]：

$$T_S(S,P,Q,N,K)=2SQP(N-K+1)^2N^2+SQPN^2K^2 \tag{11}$$

而在频域中某一层迭代一次所需的运算次数为

$$\begin{aligned}T_F(S,P,Q,N,K)=&\,8SQPN^2+4SQP(N-K+1)^2\\&+4(QS+PS+QP)N^2\log_2 N\\&+2(QS+PS+QP)(N-K+1)^2\\&\cdot\log_2(N-K+1)\end{aligned} \tag{12}$$

4 实验研究

实验的编程环境：Ubuntu 14.04 系统(64 位)，Intel i7-4790K 4.00 GHz CPU 和 32 GB RAM，采用 C++语言编程实现，使用 g++编译器进行编译。未采用任何硬件加速手段(比如：GPU)。 实验以 FXT 软件包[26]为基础进行编程。

4.1 1 维分裂基-$2/(2a)$ FFT 算法(a=1,2)运行时间比较

首先对 1 维分裂基-$2/(2a)$ FFT 算法($a=1,2$)的

运行时间进行比较，结果如图 1 所示，横坐标是输入的 FFT 点数取以 2 为底的对数得到的，纵坐标是实际运行的时间同样取对数得到，单位为 μs，各坐标点的值都是经过 100 次计算取均值得到的。由图 1 可以看出：分裂基 −2/4 算法 ($a=2$) 比基 -2 FFT 算法 ($a=1$) 运行时间更短。注意：用于测试的 1 维信号通过随机方式生成。

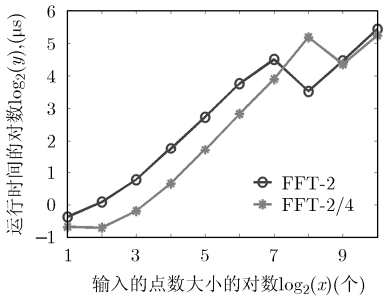


图 1 1 维分裂基-2/(2a) FFT算法(a=1,2) 实际耗费时间对比

4.2 2 维基于分裂基-2/(2a) FFT 的行列算法运用于卷积神经网络性能的比较

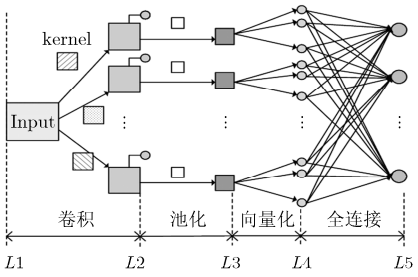
4.2.1 MNIST 手写数字分类数据集　MNIST 手写数字数据集[27]一共有 60000 幅训练图像和 10000 幅测试图像。原始图像大小为 28×28 的灰度图像，为了有效地利用 FFT 算法，我们将原始图像周围补零成为 32×32 的灰度图像。

实验的主要目的并不是要比较识别正确率，所以实验中采用的是比较浅层的卷积神经网络结构。本实验采用 5 层的卷积神经网络，其结构如图 2(a)所示。对于 MNIST 数据集大体结构如下：第 1 层为输入层；第 2 层为卷积层，在得到最终的卷积输出前还需要加上如图 2(a)所示的偏置(由小圆圈表示)，第 1 层与第 2 层之间共有 112 个连接，选用大小为 9×9 的卷积核，每个卷积核在初始化时都不一样，当然在变换到频域乘积之前需要补零到与输入图像相同的大小；第 3 层为池化(Pooling)层，池化大小为 6×6，采用最大池化方式；第 2 层与第 3 层之间采用一对一的连接方式，因此一共是 112 个连接；第 4 层是向量化层，即将第 3 层得到的小尺寸的特征图像例如 4×4 的图像，按照光栅扫描的顺序向量化为一个 16×1 的向量，将所有这样的小图像进行向量化并且连接在一起形成一个大尺寸的向量；最后一层是输出层，一共有 10 个神经元，这些神经元与第 4 层都是全连接的。

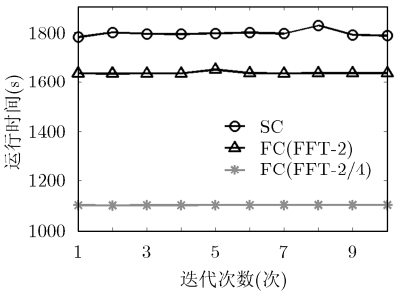
另外，隐含层即第 2 层卷积层使用了修正线性单元技术(ReLU)进行了校正，输出层使用 softmax 函数将结果归一化到 0∼1 之间。误差准则采用的是交叉熵最小准则。训练使用反向传播算法，一共迭代 10 次，训练时间结果取平均值。这些都是在卷积神经网络中关于分类问题的典型做法。

首先，作为比较的标准，在第 2 个卷积层我们使用传统的空间域直接卷积的方法实现。训练使用反向传播算法，对于 MNIST 数据集，一共迭代 10 次，训练时间结果取平均值。其次，我们使用了上文所给出的两种 2 维分裂基 -2/(2a) FFT 算法 ($a=1,2$)，实现图 2(a)中的卷积过程。在整个训练测试过程中，程序运行良好，系统能够有效的识别输入的数据集。图 2(b)给出了两种方式下 CPU 单线程运行训练过程所耗费的时间。而表 2 给出了原始图像经过补零成大小为 32×32 的输入图像后采用 3 种实现方式需要的训练时间以及训练集和测试集准确率，其中 SC 表示 2 维空域卷积，FC 表示频域卷积，FC(FFT-2)表示使用基-2 方法实现，FC(FFT-2/4)表示使用分裂基-2/4 方法实现。

从图 2(b)可以看出，对于 MNIST 数据集，在频域中实现空域线性卷积的时间优势已经显现出来，使用基-2，分裂基-2/4 的方式进行卷积的时间都比相应的空域中计算要少，这与理论分析的结果接近，同时，我们可以看到在频域实现中，使用分裂基-2/4 方式的平均训练时间最短(1102.41 s)，相



(a) 5层的卷积神经网络结构



(b) 3种线性卷积实现方法运用于卷积神经网络所耗费的时间(MNIST数据集)

图 2 实验采用的网络结构及实验结果

表 2 3 种卷积实现方式在卷积神经网络中的表现(MNIST)

卷积方式	训练精度(%)	测试精度(%)	训练时间(s)
空域卷积	96.54	96.95	1794.24
基-2 FFT	96.54	96.96	1636.45
分裂基-2/4	96.54	96.96	1102.41

较于空域实现方法(1794.24 s)，速度提升 38.56%。基-2 算法平均耗时 1636.45 s，相较于空域实现方法速度提升 8.79%。

4.2.2 Cifar-10 对象识别数据集 Cifar-10 数据集[28]一共有 10 类物体，每一类都有 60000 张图像(包括 50000 幅训练图像和 10000 幅测试图像)，每幅图像都是大小为 32×32×3 的三通道 RGB 彩色图像。同 MNIST 数据集的结果一样，使用分裂基-2/(2a)的实验结果仍然比空域中直接进行线性卷积的结果要好，详细结果如表 3 所示。

在空域卷积实现方式下，每次迭代平均训练时间约为 590 s，而使用基-2 FFT 和分裂基-2/4 FFT 实现方式的平均迭代时间分别降低为 473 s 和 343 s，相较于空域实现的方式，两者分别提速 24.73%和 72.01%。迭代 60 次后，测试集的识别率没有显著差异，并且在每次的迭代过程中，也没有显著差异。

从表 2 以及表 3 中的每一次迭代结果对比来看，本文提出的基于分裂基-2/(2a) FFT 算法的卷积神经网络实现方式在实际中并不影响网络的训练精度和测试精度。理论上来说，除了在 FFT 变换时可能有的浮点数计算时的截断误差，其余的运算从结果上与 CNN 的空域卷积实现都是等价的。结合 3.3 节中的计算复杂度的分析，假设每一层的参数简化为 $S=100$, $P=Q=32$, $N=32$, $K=5$，由式(24)和式(25)容易得到 $T_F < T_S$。目前流行的卷积神经网络以及个性化定制的网络结构仍有很多是基于“积木式”的堆叠，因此一般可以认为对于多个卷积层，其复杂度关系不变，于是该方式可以拓展到更深层的网络中。综合所有的结果来看，用分裂基-2/4 的方式代替空域中的卷积会取得最好的提速效果。

从这些实验结果来看，在频域中实现卷积神经网络中的线性卷积操作，对于最后的分类结果并不会有显著的影响。对于大小为 32×32 的输入图像，在频域中实现的方法相较于空域直接实现的方法的时间优势已经显现出来。而且随着输入图像尺寸的增大，频域实现的计算时间比空域实现要更少，因为实现 FFT 只需要 $O(M\log_2 N)$的计算复杂度，而计算空域线性卷积则需要 $O(N^2)$ 的计算复杂度。因此在 CPU 环境下利用 FFT 在频域中实现卷积神经网络能比空域直接实现线性卷积神经网络具有更快的速度。

4.2.3 基于滑动窗卷积神经网络的图像检测的应用

通用的 FFTW 软件包[23]和 FFTE 软件包[24]虽然也可以用于卷积神经网络的加速，但是因为 FFTW 和 FFTE 软件包通常都采用了较为复杂的寻优策略，不容易在小幅度修改原有快速算法结构的情况下进一步加速卷积神经网络的实现，比如：基于滑动窗卷积神经网络的图像检测的应用[25]。本文提出的基于时域抽取的分裂基-2/(2a)算法，因为采用的是一种简单的统一的快速算法结构，利用滑动窗 DFT 算法[19, 20]的策略只需要简单的注释掉输入模块中的部分代码，即可适用于滑动窗 DFT 算法的计算。

我们将在基于滑动窗卷积神经网络的图像匹配的应用场合中对比提出的基于时域抽取的分裂基-2/(2a)算法与 FFTW 对于卷积神经网络的加速性能。使用 MNIST 数据库中的部分数据构成一幅图像，如图 3 所示。图像检测的任务是在这幅大小为 320×320 的大图像中检测出所有图像为数字“0”的小块，使用的卷积网络为 4.2.1 节中已经训练好的卷积神经网络。用一个大小为 32×32 的滑动窗分别遍历图 3 中图像的每一个像素，将得到的每一幅大小为 32×32 的图像输入图 2 所示的卷积神经网络。卷积的计算分别采用所提出的基于时域抽取的分裂基-2/4 算法和 FFTW 算法[23]。图像检测的速度如表 4 所示。

表 3 不同卷积方式在 Cifar-10 数据集上的迭代结果(后 5 次)

迭代次数	训练集精度(%)			测试集精度(%)			训练时间(s)		
	空域	基-2	基-2/4	空域	基-2	基-2/4	空域	基-2	基-2/4
56	69.41	69.34	69.37	65.85	65.81	65.82	590.650	473.986	343.576
57	69.33	69.24	69.27	65.79	65.80	65.81	587.991	473.982	343.607
58	69.34	69.28	69.28	65.94	65.90	65.87	687.968	473.991	343.578
59	69.35	69.31	69.36	65.94	65.96	65.81	588.337	473.998	343.572
60	69.56	69.50	69.53	66.03	65.91	65.90	588.126	473.971	343.571

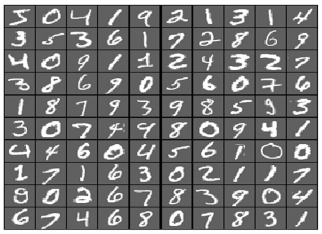


图 3　MNIST 数据库中的部分图像

表 4 基于滑动窗卷积神经网络的图像检测速度比较(s)

图像	基于时域抽取分裂基-2/4 算法加速的滑动窗卷积神经网络图像检测耗费总时间	基于FFTW加速的滑动窗卷积神经网络图像检测耗费总时间
MNIST 部分图像	187.73	205.78

由表 4 可以看出：在基于滑动窗卷积神经网络的图像检测的应用中，本文提出的基于时域抽取分裂基-2/4 算法加速性能比 FFTW 速度提高大约 8.7%。

5　结论

本文首先提出了一种统一的基于时域抽取方法的分裂基-2/(2a)1 维 FFT 快速算法，其中 a 为任意自然数，然后通过行列算法将提出的 1 维 FFT 算法扩展到 2 维。在 CPU 环境下对提出的 FFT 算法在一种卷积神经网络中的加速性能进行了研究。在 MNIST 手写数字数据库以及 Cifar-10 对象分类中的实验结果表明：在不损失训练精度和测试精度的情况下利用分裂基-2/4 FFT 算法和基-2 FFT 算法实现的卷积神经网络比空域直接实现的卷积神经网络分别提速 38.56%, 8.79%和 72.01%, 24.73%。因此，在频域中实现卷积神经网络的线性卷积操作是一种十分有效的实现方式。

参 考 文 献

[1] HINTON G E and SALAKHUTDINOV R R. Reducing the dimensionality of data with neural networks[J]. *Science*, 2006, 313: 504–507. doi: 10.1126/science.1127647.

[2] HINTON G E, OSINDERO S, and TEH Y W. A fast learning algorithm for deep belief nets[J]. *Neural Computation*, 2006, 18(7): 1527–1554. doi: 10.1162/neco.2006.18.7.1527.

[3] BENGIO Y, COURVILLE A, and VINCENT P. Representation learning: A review and new perspectives[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013, 35(8): 1798–1828. doi: 10.1109/TPAMI.2013.50.

[4] LECUN Y, BENGIO Y, and HINTON G E. Deep learning[J]. *Nature*, 2015, 521(7553): 436–444. doi: 10.1038/nature14539.

[5] DENG L and YU D. Deep learning: Methods and applications[J]. *Foundations and Trends in Signal Processing*, 2014, 7(3): 197–387.

[6] LECUN Y, BOTTOU L, BENGIO Y, *et al*. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324. doi: 10.1109/5.726791.

[7] KRIZHEVSKY A, SUTSKEVER I, and HINTON G E. Imagenet classification with deep convolutional neural networks[C]. Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 2012: 1097–1105.

[8] SZEGEDY C, LIU W, JIA Y Q, *et al*.. Going deeper with convolutions[C]. IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 2015: 1–9.

[9] JADERBERG M, VEDALDI A, and ZISSERMAN A. Speeding up convolutional neural networks with low rank expansions[J]. *Computer Science*, 2014, 4(4): XIII.

[10] LIU B, WANG M, FOROOSH H, *et al*. Sparse Convolutional neural networks[C]. IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, United States, 2015: 806–814.

[11] VASILACHE N, JOHNSON J, MATHIEU M, *et al*. Fast convolutional nets with fbfft: A GPU performance evaluation [C]. International Conference on Learning Representations, San Diego, CA, USA, 2015: 1–17.

[12] COOLEY J W and TUKEY J W. An algorithm for the machine calculation of complex Fourier series[J]. *Mathematics of Computation,* 1965, 90(19): 297–301. doi: 10.2307/2003354.

[13] DUHAMEL P and VETTERLI M. Fast Fourier transforms: A tutorial review and a state of the art [J]. *Signal Processing*, 1990, 19(4): 259–299. doi: 10.1016/0165-1684(90)90158-U.

[14] WINOGRAD S. On computing the discrete Fourier transform[J]. *Proceedings of the National Academy of Sciences of the United States of America*, 1976, 73(4): 1005–1006. doi: 10.1073/pnas.73.4.1005.

[15] KOLBA D P and PARKS T W. A prime factor algorithm using high-speed convolution[J]. *IEEE Transactions on Acoustics Speech & Signal Processing*, 1977, 25(4): 281–294. doi: 10.1109/TASSP.1977.1162973.

[16] DUHAMEL P and HOLLMANN H. Implementation of Split-radix FFT algorithms for complex, real, and real symmetric data[C]. IEEE International Conference on Acoustics, Speech, and Signal Processing, Tampa, FL, USA, 1985: 285–295.

[17] BOUGUEZEL S, AHMAD M O, and SWAMY M N S. A general class of split-radix FFT algorithms for the computation of the DFT of length-2^m [J]. *IEEE Transactions on Signal Processing,* 2007, 55(8): 4127–4138. doi: 10.1109/

TSP.2007.896110.

[18] BI G, LI G, and LI X. A unified expression for split-radix DFT algorithms[C]. IEEE International Conference on Communications, Circuits and Systems, Chengdu, China, 2010: 323–326.

[19] FARHANG BOROUJENY B and LIM Y C. A comment on the computational complexity of sliding FFT[J]. *IEEE Transactions on Circuits and Systems II Analog and Digital Signal Processing*, 1992, 39(12): 875–876. doi: 10.1109/82.208583.

[20] PARK C S and KO S J. The hopping discrete Fourier transform[J]. *IEEE Signal Processing Magazine*, 2014, 31(2): 135–139. doi: 10.1109/MSP.2013.2292891.

[21] GOUK H G and BLAKE A M. Fast sliding window classification with convolutional neural networks[C]. Proceedings of the 29th International Conference on Image and Vision Computing, New Zealand, 2014: 114–118.

[22] RAO K R, KIM D N, and HWANG J J. Fast Fourier Transform: Algorithms and Applications[M]. Berlin: Springer Science & Business Media, 2011: 5–6.

[23] FRIGO M and JOHNSON S G. FFTW: An adaptive software architecture for the FFT[C]. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, WA, USA, 1998: 1381–1384.

[24] TAKAHASHI D. FFTE: A fast fourier transform package [OL]. http://www.ffte.jp/, 2014.2.

[25] SERMANET P, EIGEN D, ZHANG X, *et al*. Overfeat: Integrated recognition, localization and detection using convolutional networks[OL]. arXiv:1312.6229. 2013: 1–16.

[26] ARNDT J. Fxtbook[OL]. http://www.jjj.de/fxt/#fxtbook. 2015.1.

[27] LECUN Y, CORTES C, and CHRISTOPHER J C. The MNIST database of handwritten digits, Burges[OL]. http://yann.lecun.com/exdb/mnist/, 2015.5.

[28] KRIZHEVSKY A, NAIR V, and HINTON G. Cifar-10 dataset[OL]. http://www.cs.toronto.edu/~kriz/cifar.html, 2009.1.

伍家松： 男，1983 年生，讲师，研究方向为卷积网络、离散正交变换快速算法.

达 臻： 男，1992 年生，硕士生，研究方向为卷积网络.

魏黎明： 女，1993 年生，硕士生，研究方向为卷积网络.

SENHADJI Lotfi： 男，1966 年生，教授，研究方向为生物医学信号处理.

舒华忠： 男，1965 年生，教授，研究方向为信号与图像处理.