Bilinear algorithms for convolution

A survey of convolution algorithms and their stability

Caleb Ju & Edgar Solomonik

University of Illinois at Urbana-Champaign

Abstract

The recent popularity of convolutional neural networks have motivated the study of convolution without using the fast Fourier transform. While alternatives, like the Winograd minimal filtering and Toom-Cook method, have been proposed, there is little work in the mathematical definition and numerical analysis of these algorithms. In this survey, we use the framework of bilinear algorithms to introduce the various methods. We provide novel stability bounds and propose methods to mitigate the error.

Introduction

Convolution:

- Calculates the interaction of two functions to create a third function
- Used in integer multiplication, signal processing, and solutions to PDEs
- Key component ($\leq 90\%$ runtime [1]) in convolutional neural networks (CNN)

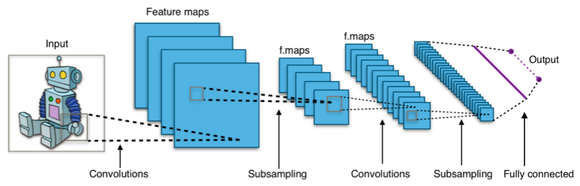

Figure 1: CNN Architecture

Unlike other problems, CNN's use of convolution has:

- Small to medium filter sizes
- Many convolutions for one image
- Larger errors can be tolerated depending on CNN design

Blackbox algorithms today are,

Operation	Pros	Cons
Direct method	Highly optimized, Easy to implement	No savings
Matrix multiplication	Highly optimized	Large dimensions, Matrix multiplication
FFT	Asymptotically fast, Good for large filters	Complex arithmetic
Winograd's minimal filtering [2]	Savings with small filters	Unknown overhead, Errors if large filter

Main Objectives

1. Introduce algorithms via bilinear algorithms, compare pros and cons
2. Bound and reduce errors of various algorithms

Convolution as a bilinear algorithm

Discrete linear convolution: given inputs $f \in \Re^b$ and $g \in \Re^n$, their convolution is computed by

$$y_k = \sum_{i=0}^{b-1} f_i g_{k-i}. \tag{1}$$

Can be computed by Hankel matrix multiplication,

$$y = \begin{bmatrix} 0 & \dots & 0 & 0 & f_0 \\ 0 & \dots & 0 & f_0 & f_1 \\ \vdots & \dots & \dots & \dots & \vdots \\ f_0 & \dots & \dots & \dots & f_{n-1} \\ \vdots & \dots & \dots & \dots & \vdots \\ f_{n-1} & 0 & \dots & \dots & 0 \end{bmatrix} \begin{bmatrix} g_{n-1} \\ g_{n-2} \\ \vdots \\ \vdots \\ \vdots \\ g_0 \end{bmatrix}. \tag{2}$$

We can cast the problem as a *bilinear algorithm*,

$$y = S(f,g) = C\Big(Af \odot Bg\Big). \tag{3}$$

1. **Toom-Cook method**.
   - *Interpolation* problem. Evaluate $\rightarrow$ evaluate $\rightarrow$ multiply $\rightarrow$ interpolate.
   - Set matrices A,B, and C to Vandermonde matrices and its inverse,

$$V_{(p)} = \begin{bmatrix} p_0^0 & p_0^1 & \dots & p_0^{n-1} \\ p_1^0 & p_1^1 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ p_{n-1}^0 & \dots & \dots & p_{n-1}^{n-1} \end{bmatrix} \tag{4}$$

   using nodes $p = \begin{bmatrix} 0 & 1 & -1 & \dots \end{bmatrix}$.
   Advantage: Easy to derive, good for small filters
   Disadvantage: Expensive and inaccurate if filters too large
2. **Winograd minimal filtering method**. Like Toom-Cook, except:
   - Evaluate remainder instead of full polynomials
   - Use Chinese Remainder Theorem instead of interpolation

   Advantage: Cheaper overhead of evalutation and interpolation
   Disadvantage: Difficult to derive. As inaccurate as Toom-Cook
3. **r-tap algorithms**. Useful when the input is larger than filter. Given convolution algorithm $y = C(Af \odot Bg)$, the respective r-tap algorithm is

$$y = A^T\Big((C^T f) \odot (Bg)\Big). \tag{5}$$

   The representation is a full-Hankel matrix, i.e. no zero terms,

$$y = \begin{bmatrix} g_0 & g_1 & g_2 & g_3 \\ g_1 & g_2 & g_3 & g_4 \\ g_2 & g_3 & g_4 & g_5 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}. \tag{6}$$

   Same advantages and disadvantages as Toom-Cook
4. **Discrete and Fast Fourier Transform**. Toom-Cook with nodes $p = [\omega_n^i]$ for $i = 0, 1, \dots, n-1$.
   Advantage: Easy to derive, $O(n\log(n))$, accurate
   Disadvantage: Expensive for small filters

Bounds and Reduction of Error

To reduce error, improve conditioning of the matrices. We propose:

1. Break convolution into short ones by Kronecker products (nested)
2. Better nodes, like Chebyshev nodes $p_i = \cos(\frac{2i-1}{2n}\pi)$
3. Use orthogonal basis like Chebyshev basis. Has unconditional stability

Method	Error Bounds
Toom-Cook	$\Omega\Big((\frac{n-1}{2})^n/\sqrt{n}\Big)$
Nested Toom-Cook	$\Omega(n^{k\log_k([k-1]/2)-1/2})$
Toom-Cook Chebyshev Nodes	$\Omega\Big(\frac{(1+\sqrt{2})^n}{\sqrt{n}}\Big)$
Toom-Cook Chebsyehv Basis	$O(1)$
DFT/FFT	$O(1)$

Table 1: Error Bounds

Experimental Results

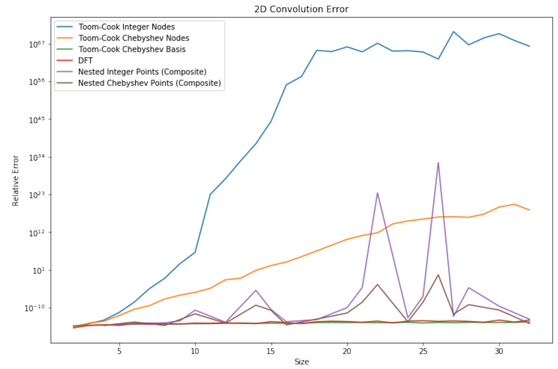

Figure 2: Error of convolution on random 2D matrices

From the graph, we can see it supports our original bounds.

1. Long convolutions broken down to a series of many short convolutions are very stable
2. Chebyshev nodes can extend the use of the Vandermonde matrices
3. Toom-Cook is unusable after sizes of eight, and the Chebyshev basis is unconditionally stable

Concluding Remarks and Future Directions

- CNNs have efficient algorithms for small filters (Winograd and Toom-Cook) and large filters (FFT), however the decision is non-trivial for medium size filters
- Use of general algorithms can help design more stable ones for any input size and help compare speed and accuracy trade-offs
- For future works, implement these algorithms for 2D and 3D convolution algorithms on supercomputers and/or GPUs and derive communication bounds for nested bilinear algorithms

References

[1] Jason Cong and Bingjun Xiao. Minimizing computation in convolutional neural networks. International Conference on Artificial Neural Networks, 2014.

[2] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. The IEEE Conference on Computer Vision and Pattern Recognition, 2016.