HUAWEI APPLIED MATHEMATICAL CONTEST

Contest Thesis

Low Dissipation and Self-adapting FIR Filter Algorithms

Author:
Chai Hao,
Li Minghao,
Wang Xiyuan

Team:
REK3000

A thesis submitted in fulfillment of the requirements
for the HUAWEI Applied Mathematics ContestApplied Mathematics Contest

in August 2023

Fudan University, Nanjing University
Chai Hao in SMCS,Fudan University,
Li Minghao in EIE,Nanjiing University,
Wang Xiyuan in SMS,Fudan University

华为领航杯应用数学大赛论文

低功耗自适应有限脉冲滤波器算法

2023.8

摘要

本文主要实现一种低功耗 (Low Dissipation) 自适应 (Self-Adjusting) 的有限长单位脉冲响应滤波器 (Finite Impulse Response Filter) 算法。主要参考文献是 [Winograd 1968; Gholami et al. 2021; Jiang et al. 2020]。

其中低功耗算法的主要思想是降低滤波器的乘法器的运算复杂度。主流方法有 [Winograd 1968] 中的内积加速算法和利用机器学习方法优化滤波器参数，参考[Gholami et al. 2021]。

自适应算法采用在线算法，主要解决保证低延迟的问题。

滤波器实现了经典的 Cooley-Tukey 构造的快速傅立叶变换算法 (Fast Fourier Transformatioin, FFT)。实现细节参考附录或Wikipedia。

本文是第一届华为领航杯应用数学大赛参数论文并依照该竞赛要求格式完成。

论文目录

1 问题背景阐述

1.1 有限脉脉冲响应滤波器 (Finite Impulse Response Filter, FIR Filter)

工程上进行 FIR 滤波器 (FIR Filter, 以下简称 FIR 或 FIR 滤波器) 是非递归型滤波器的简称，全称是有限长单位脉冲响应滤波器 (Finite Response Filter)。

其中带有常数系数的如下图

1.2 离散傅立叶变换及快速实现 (Discrete/Fast Fourier Transformatioin, DFT/FFT)

如果采用上述一般形式的离散 Fourier 变换公式在硬件上进行运算，那么完成一次完整的 DFT 需要进行 $\mathcal{O}(n^2)$ 次乘法和 $\mathcal{O}(n^2)$ 次加法。其中 n 代表 DFT 变换器的阶数 (或者滤波器的抽头数)

在 20 世纪 60 年代，来自 Princeton 的计算机科学家 Cooley 和数学家 Tukey 发明了一种基于快速幂算法和 Gauss 和算术性质的快速离散 Fourier 算法。其可以通过数量级地减少 DFT 算法中的乘法次数大大加快原有离散 Fourier 变换的运算速度。本质上这一算法的理论框架已经被德国数学家 Carl Friedrich Gauss 于 19 世纪初发现并证明，参见。

1.3 滤波器算法中的乘法器性能

1.4 技术诉求

针对华为领航杯应用数学大赛的难题 ‘‘低功耗自适应 FIR 滤波器’’ 有如下三个主要技术诉求。

1. 诉求一：通过改变滤波器架构设计等办法，使得其硬件算法整体性能提升 40%。
2. 诉求二：指定位宽下引入近似乘法，实现乘法资源降低 50%，误差或信噪比 (SNR) 大于 75dBc ；或者使用机器学习模型训练，使得其训练性能基本无损 (< 0.1dB)
3. 诉求三：" 低位宽" FIR 滤波器算法实现 (< 6bit)，并能够实现模型稳定校正且性能基本无损 (< 0.1dB)

在本文第二部分经典方法复现中，三个子节将会依次详细论述本项目对上述三个诉求的经典解决方式。

在第三部分 ‘‘理论和算法’’ 中本文将尽可能清晰地叙述项目中使用的关键算法理论，具体创新和优化的细节部分。

在第四部分 ‘‘实验数据及软硬件介绍’’ 和第五部分 ‘‘实验结果及主要结论’’ 中本文将展示根据前三部分的算法理论实现的工程在仿真环境的测试数据以及指标比较。

最后关于项目的复现以及移植，完整的源代码解释和运行环境信息全面呈现在附录一节。

2 经典方法复现

2.1 基于 Cooley-Tukey FFT 算法实现的 FIR 滤波器

具体理论可参考维基百科。

2.2 冗余计算和 Booth 乘法

Booth 乘法器的原生算法可以在保持设定精度的前提下显著地提升滤波器中乘法器的性能。这里简要介绍 Booth 乘法的实现方法。

采用静态优化技术设计合适的滤波器

设计合适的乘法位宽

分布合适数量的乘法器

2.3 基于量化感知的自适应加速

针对于由于在全精度滤波器中乘法器进行的是浮点运算

小规模硬件 (如基带芯片) 上布局神经网络要求低缓存和低延迟。这样的算法才能满足硬件上信号处理的速度和硬件设备自身的条件限制。

量化感知技术 (Quantization Perception) 可以减轻神经网络训练上浮点数计算的时间/空间复杂度，从而释放算力进而加速神经网络的训练。

对于浅层神经网络而言采用前向直接估计器 (STE)

而使用非前向直接估计器。

有关于自适应滤波器 FIR 滤波器

3 理论及算法

3.1 离散傅立叶变换 (Discrete Fourier Transformatioin, FFT) 的快速实现

快速傅立叶变换 (通常只是至离散情形的) 以下内容参考 wikipedia 。如

第一种办法：Wingoard 加速算法 (Wingoard Fourier Transformation,WFT)

第二种办法：Cooley-Tukey 算法 (Cooley-Tukey Algorithms)

此方式适用于 $n = 2^m$ 时，这里的实现方式主要参考了算法导论一书的 FFT 章节见 [Cormen et al. 2022, Polynomials and FFT]

另外更加高效的办法是参考 [Li et al. 2010]。或者关于快速傅立叶变换的综述性质书籍[Nussbaumer 1982]。

第三种办法

3.2 乘法器加速的 Booth 算法 (Booth Multiplication Algorithms)

Karatsuba 算法

以下对于高精度大整数乘法的 Karatsuba 算法简介摘录自 OI Wiki:Karatsuba Algorithm。

如果取高精度数字 (必要时同时约化为大整数) 的位数为 n，那么高精度—高精度竖式乘法需要花费 $\mathcal{O}(n^2)$ 的时间。本节介绍一个时间复杂度更为优秀的算法，由前苏联（俄罗斯）数学家 Anatoly Karatsuba 提出，是一种分治算法。

考虑两个十进制大整数 x 和 y，均包含 n 个数码并可以有前导零。任取 $0 < m < n$，记

$$x = x_1 \cdot 10^m + x_0,$$
$$y = y_1 \cdot 10^m + y_0,$$
$$x \cdot y = z_2 \cdot 10^{2m} + z_1 \cdot 10^m + z_0,$$

其中 $x_0, y_0, z_0, z_1 < 10^m$。简单四则运算可得

$$z_2 = x_1 \cdot y_1,$$
$$z_1 = x_1 \cdot y_0 + x_0 \cdot y_1,$$
$$z_0 = x_0 \cdot y_0.$$

继续观察 (本质上使用分配律或 Wingoard 加速算法) 将

$$z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$$

从而只要计算出 $(x_1 + x_0), (y_1 + y_0)$ 再与 z_2, z_0 相减即可求出 z_1

因为普遍讲计算机上乘法的耗时要大于甚至远大于加法 (尤其是数位越大的时候)。从而将乘法替换为加法会大大降低算法的渐进时间复杂度。

上式实际上是 Karatsuba 算法的核心，它将长度为 n 的乘法问题转化为了 3 个长度更小的子问题。若令

$$m = \left\lceil \frac{n}{2} \right\rceil$$

记 Karatsuba 算法计算两个 n 位整数乘法的耗时为 $T(n)$，则有

$$T(n) = 3 \cdot T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \mathcal{O}(n)$$

由主定理可得

$$T(n) = \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.585})$$

这相比于通常的大整数乘法的 $\mathcal{O}(n^2)$ 有数量级的减少。也回答了 20 世纪 60 年代 Kolmogorov 关于大整数乘法渐进时间复杂度是否为 $\Omega(n^2) \approx \mathcal{O}(n^2)$ 的重大问题。

而 Anatoly Karatsuba 发明这个算法时年仅 23 岁，仅仅是在 Kolmogorov 陈述他关于任何大整数乘法算法的时间复杂度趋于 $\mathcal{O}(n^2)$ 这一猜测一周后 Karatsuba 就完成了上述更快算法的构造。具体的历史信息可参考 Wikipedia:Karatsuba Algorithm。其中一段摘录如下

> - In 1960, Kolmogorov organized a seminar on mathematical problems in cybernetics at the Moscow State University, where he stated the $\Omega(n^2)$ conjecture and other problems in the complexity of computation. Within a week, Karatsuba, then a 23-year-old student, found an algorithm that multiplies two n-digit numbers in $\mathcal{O}(n^{\log_2 3})$ elementary steps, thus disproving the conjecture. Kolmogorov was very excited about the discovery; he communicated it at the next meeting of the seminar, which was then terminated. Kolmogorov gave some lectures on the Karatsuba result at conferences all over the world (see, for example, "Proceedings of the International Congress of Mathematicians 1962", pp. 351–356, and also "6 Lectures delivered at the International Congress of Mathematicians in Stockholm, 1962") and published the method in 1962, in the Proceedings of the USSR Academy of Sciences. The article had been written by Kolmogorov and contained two results on multiplication, Karatsuba's algorithm and a separate result by Yuri Ofman; it listed "A. Karatsuba and Yu. Ofman"as the authors. Karatsuba only became aware of the paper when he received the reprints from the publisher.

更完整的第一手算法理论参考可见 Karatsuba 本人的综述 [Karatsuba 1995]。本项目使用其思想实现了针对滤波器乘法的版本如下，主要集成了 Karatsuba 算法实现了多项式乘法和卷积，最后再处理所有的进位问题。

3.3 量化感知中的混合精度量化 (Mixed Precision Quantization)

此部分主要参考 "A Survey of Quantization Methods for Efficient Neural Network Inference" 的大量参考文献中实现的不同的量化感知技术。其中

在神经

在这篇文章中

4 实验数据及软硬件介绍

4.1 基于昇腾/ARM 架构的 xxx 型号芯片的仿真测试数据

4.2 基于 OpenNN/PyTorch 框架的量化感知算法测试数据

以下是神经网络的测试算法。

4.3 实验用软硬件信息

5 实验结果及主要结论

5.1 主要测试结果

5.2 结论

实现性能 40% 的提升的结论。

6 总结及展望

有限长单位脉冲响应滤波算法是一种广泛应用在通信工程、音视频处理、信号处理等工程方向的硬件算法。其作为信号处理技术中的核心主流算法，已经被多系列主流基带芯片通信芯片所集成。如何高效执行有限长单位脉冲响应滤波算法，无论是从算法理论上还是硬件兼容软件集成上都是学界和工业界的主攻难题。

量化感知是近二三十年来发展迅速的加速神经网络训练的主流技术之一。从提出至今已经有不同研究者提出了种类繁多的量化策略其想要解决的主要问题是在一些性能限制下最小化如下的量化后数据和原始数据的偏差

$$\min \sum_{r} \|Q(r) - r\|_2, \qquad \text{s.t.} \quad \text{Constraints(r)} = 0$$

高效的量化感知算法可以大大加快神经网络训练的底层运算速度，是一个加速神经网络训练的可行方向。但是另一方面，神经网络自身的结构优劣和误差传播算法的设计好坏也是制约其训练质量和速度的重要指标。从这一意义上说，设计更好的神经网络结构和与之配套的误差传播/激励传播算法可能会从另一个方向优化神经网络的训练。这种优化可能达到量化感知技术无法实现的程度。

综观工程技术发展历史，最底层的算法和技术迭代往往艰深险难，每进步一毫可能都要耗费巨大的时间人力成本，更甚者算法层面的进步可能需要数学理论的艰难创新或是天才般的灵光乍现。本文集成了快速傅立叶变换算法、低功耗约束的高效规划算法、量化感知加速技术等诸多经典或者前沿的工程算法，其中不妨极具影响力的 '世界级' 算法。经过软硬件整合实现一套可以在华为系硬件芯片上运行的完整的自适应低功耗有限长单位脉冲响应滤波算法 (Self-Adjusting Low-Dissipation Finite Impulse Response Filter)

附录中包含了可执行的 C++ 工程的源代码实现以及详细注释；量化感知技术实现的神经网络框架 (基于 OpenNN/Tensorflow) 源代码实现和接口注释；以及详细的操作方式。(API Source Code;Description;Manual)

如果有后续的经费支持，我们团队认为我们可能继续维护该项目的开发，并且生产出更加高效的算法。

附录

这是附录文件。

6.1 源程序文件树

全工程文件树如下图所示

```
HUAWEI_FIR
├─ ACFIR
│   └─ ...
├─ MLPyPlugin
│   └─ ...
├─ VDHL
│   └─ ...
└─ etc
```

其中 ACFIR 文件夹是可运行在硬件或仿真系统上的 FIR 滤波算法源代码；MLPyPlugin 文件夹是优化滤波器性能的机器学习 Python 代码文件；VDHL 是用于仿真模拟或硬件测试的硬件描述语言源代码。“etc” 表示其他诸如测试数据，日志文件，编译软件生成文件等附加的文件。“...” 表示文件夹中存在的没有列举出的文件。

关于本论文和后续汇报的幻灯片文件将在 GitHub 团队 地址 上的 Contest-Piece 仓库中的 “华为领航杯论文” 文件夹中发布。由于自身嵌套问题此文件恕不在附录的源程序文件解释部分解释，也于文件树上省略此文件夹。

同时此比赛的一切源代码均发布到指定邮箱。

>>> 此处添加 ACFIR 文件树

```
ACFIR
├─ benchmark.h
├─ benchmark.cpp
└─ ...
```

程序解答

transformation.h 这是表示引用文件。

```
    class HelloWorldApp {
        public static void main(String[] args) {
            System.out.println("Hello World!"); // Display the string.
            for (int i = 0; i < 100; ++i) {
                System.out.println(i);
            }
        }
    }
```

>>> 此处添加 Python 机器学习框架文件库

MLPyPlugin
- data_config.py
- torch_config.py
- run_strategy.wheel
- ...

>>> 其他

程序解读

6.1.1 滤波器代码注解

下面的 C++ 函数实现了滤波器的基准测试功能。

使用信号处理的名字空间

convolution.cpp

```
    #include "convolution.h"

    namespace dsplib{
        namespace conv{
            vfp fir_regular(const vfp& seq, const vfp& tap) {
                int slen = seq.size();
                int tlen = tap.size();
                int olen = slen + tlen - 1;
                vfp vo(olen);
                for(int i = 0; i < olen; i++) {
                    vo[i] = 0;
                    for(int t = 0; t <= i; t++) {
                        if((t < tlen) && ((i - t) < slen))
                            vo[i] += tap[t] * seq[i - t];
                    }
                }
                return vo;
            };
        }
    }
```

其中等等。

6.1.2 机器学习框架代码注解

此时使用 torch 包对滤波器参数选择进行神经网络优化。参考

```
        from __future__ import division
        import gym
        import numpy as np
        import torch
        from torch.autograd import Variable
        import os
        import psutil
        import gc
        import random
        import train
        import buffer
        import matplotlib.pyplot as plt
        import pandas as pd
        from tqdm import tqdm
        import sys
```

6.1.3 可视化框架注解

这里使用

6.2 使用指南

6.2.1 操作系统和软件要求

6.2.2 执行步骤

>>> run xxx.exe
拟代码 (pseudo code)
StepA, StepB, ... C ....

参考文献

Cormen2022 Thomas H Cormen et al. *Introduction to algorithms*. 4th ed. MIT press, 2022 (cit. on p. 6).

Gholami2021 Amir Gholami et al. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: (Mar. 2021). arXiv: 2103.13630 [cs.CV] (cit. on pp. 2, 7).

Jiang2020 Honglan Jiang et al. "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications". In: *Proceedings of the IEEE* 108.12 (2020), pp. 2108–2135. DOI: 10.1109/JPROC.2020.3006451 (cit. on p. 2).

Karatsuba1995 Anatolii Alexeevich Karatsuba. "The complexity of computations". In: *Proceedings of the Steklov Institute of Mathematics-Interperiodica Translation* 211 (1995), pp. 169–183 (cit. on p. 7).

Li2010 Chao Li et al. *Mathematical Principle of Computer Algebra System*. TsingHua University Press, 2010. ISBN: 9787302230106. URL: https://mathmu.github.io/MTCAS/mtcas.pdf (cit. on p. 6).

Nussbaumer1982 Henri J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Berlin Heidelberg, 1982. DOI: 10.1007/978-3-642-81897-4 (cit. on p. 6).

Winograd1968 S. Winograd. "A New Algorithm for Inner Product". In: *IEEE Trans. Comput.* 17.7 (June 1968), pp. 693–694. ISSN: 0018-9340. DOI: 10.1109/TC.1968.227420. URL: https://doi.org/10.1109/TC.1968.227420 (cit. on p. 2).