УДК 004.932

УСКОРЕНИЕ ВЫЧИСЛЕНИЯ РАЗРЕЖЕННОГО ПРЕДСТАВЛЕНИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ JIT-КОМПИЛЯЦИИ

А.И. Лепаловская[1]

[1]*Самарский университет, г. Самара*
e-mail: arina.lepalovskaya@mail.ru

Аннотация

В работе рассматривается задача оптимизации вычислительной сложности алгоритма итерационного мягкого порога (ISTA) для восстановления изображений. Основное внимание уделено переходу от интерпретируемой реализации на языке Python к компилируемому коду с использованием библиотеки Numba. Экспериментально доказана эффективность применения JIT-компиляции и распараллеливания вычислений, что позволило сократить время обработки данных при сохранении высокого качества восстановления.

Ключевые слова: разреженное представление, ISTA, Numba, JIT-компиляция, оптимизация кода, параллельные вычисления.

Введение

Современные задачи компьютерного зрения, такие как восстановление поврежденных областей и шумоподавление, часто базируются на методе разреженных представлений. Данные подходы показывают высокую эффективность в задачах фильтрации аддитивного шума и восстановления поврежденных областей [1]. Суть метода заключается в аппроксимации фрагмента изображения малым числом элементов избыточного словаря. Реализация данных подходов сталкивается с проблемой низкой производительности при обработке полноцветных изображений высокого разрешения. В данной работе исследуется метод ускорения таких вычислений с использованием библиотеки Numba.

1. Постановка задачи и базовый алгоритм

В данной работе решается задача разреженного представления изображения по избыточному словарю с ограничением на число ненулевых коэффициентов. Основой метода является выражение:

$$y = Dx, \tag{1}$$

где $y \in R^N$ – вектор исходного фрагмента изображения, $D \in R^{N \times K}$ – избыточный словарь $(K > N)$, x – искомый разреженный вектор коэффициентов [2].

В рамках проводимого исследования изображение разбивается на блоки размером 8×8 пикселей, что определяет размерность вектора $N = 64$. Для обеспечения избыточности используется косинусный словарь $D \in R^{64 \times 441}$, состоящий из $K = 441$ атома.

Поиск вектора коэффициентов x для каждого блока изображения сводится к минимизации функционала:

$$F(x) = \frac{1}{2}\|y - Dx\|_2^2 + \lambda\|x\|_1, \quad (2)$$

где λ – параметр регуляризации, который определяет баланс между точностью восстановления и степенью разреженности.

Алгоритм ISTA реализует итерационный процесс поиска x по формуле:

$$x_k = S_{\eta\lambda}(x_{k-1} - \eta D^T(Dx_{k-1} - y)), \quad (3)$$

где S – оператор мягкой пороговой обработки, а η – шаг обучения, определяемый спектральной нормой матрицы словаря.

2. Оптимизация вычислений средствами Numba

Библиотека Numba позволяет выполнять JIT-компиляцию (Just-In-Time) кода Python, что особенно эффективно для итерационных алгоритмов, состоящих из простых арифметических операций над массивами [3]. Внутренний цикл функции восстановления реализует следующий пошаговый алгоритм:

1. Вычисление текущего градиента: $g = D^T(Dx - y)$.
2. Обновление вектора коэффициентов: $x_{temp} = x - \eta \cdot g$.
3. Применение пороговой фильтрации: $x = sign(x_{temp}) \cdot \max(|x_{temp}| - \eta\lambda, 0)$.

Поскольку данные операции повторяются K (максимальное число итераций) раз для каждого из блоков изображения, трансляция Python-кода в машинные инструкции позволяет минимизировать временные затраты на работу интерпретатора Python. В процессе программной реализации были использованы следующие подходы для повышения производительности:

1. Использование JIT-компиляции (Just-In-Time) с помощью декоратора @njit. Это позволило транслировать код функций в машинные инструкции, минуя Global Interpreter Lock (GIL) [3].

2. Внедрение критерия ранней остановки. Если норма вектора разности коэффициентов между итерациями $\|x_k - x_{k-1}\|_2^2$ становится меньше порога $tol = 10^{-4}$, то цикл прерывается. Это позволяет экономить вычислительные ресурсы на однородных участках изображения.

3. Распараллеливание внешнего цикла обработки каналов и блоков с помощью параметра parallel=True и функции prange, что обеспечило равномерную загрузку всех ядер центрального процессора.

3. Результаты экспериментов

Тестирование проводилось на тестовом изображении с наложенным импульсным шумом с использованием процессора AMD Ryzen 5 1500X. Результаты сравнительного анализа времени выполнения алгоритмов представлены в таблице 1. Стоит отметить, что JIT-компиляция требует дополнительных затрат времени при первом вызове функции для генерации машинного кода, однако при обработке последующих блоков или изображений достигается максимальный выигрыш в скорости.

Таблица 1 – Сравнение времени выполнения алгоритма

Метод реализации	**Время обработки (сек)**
NumPy (базовый)	1112.31
Numba (JIT + Parallel)	295.60

Стандартный шаг обработки, равный размеру блока (шаг = 8), вызывает появление видимых границ между блоками, как показано на рисунке 1. Использование уменьшенного шага (шаг = 2) с перекрытием и последующим усреднением фрагментов позволило устранить эти артефакты и обеспечить визуальную целостность изображения. Результат восстановления представлен на рисунке 2.

а) б)

Рисунок 1 – Пример восстановления зашумленного изображения ($\lambda = 0.15, K = 200,$ шаг $= 8$):
а) зашумленное изображение, б) изображение, восстановленное с помощью словаря

а) б) в)

Рисунок 2 – Пример восстановления зашумленного изображения ($\lambda = 0.15, K = 200,$ шаг $= 2$):
а) оригинальное изображение, б) зашумленное изображение,
в) изображение, восстановленное с помощью словаря

Заключение

В ходе работы была выполнена программная реализация алгоритма вычисления разреженного представления изображений. Применение JIT-компиляции Numba позволило значительно уменьшить время вычислений, сохранив гибкость разработки на языке Python. Экспериментально подтвержденное ускорение процесса восстановления открывает возможности для применения сложных методов разреженного анализа в задачах, близких к реальному времени.

Список использованных источников

1. Архипова, Д. В. Исследование метода шумоподавления на основе разреженного представления / Д. В. Архипова, Е. В. Гошин // Информационные технологии и нанотехнологии (ИТНТ-2023): сборник трудов по материалам IX Международной конференции и молодежной школы, Самара, 17-23 апреля 2023 года. – Самара: Издательство Самарского университета, 2023. – Т. 3. – С. 85–86.

2. Rubinstein, R. Dictionaries for Sparse Representation Modeling / R. Rubinstein, A. M. Bruckstein, M. Elad. // Proceedings of the IEEE. – 2010. – V. 98, № 6. – P. 1045–1057.

3. Lam, S. K. Numba: A llvm-based python jit compiler / S. K. Lam, P. Pitrou, S. Seibert // Proceedings of the Second Workshop on the Python for High-Performance Computing Pyramid. – 2015. – P. 1–6.

ACCELERATING SPARSE REPRESENTATION COMPUTATION FOR IMAGES USING JIT COMPILATION

A.I. Lepalovskaya[1]

[1]Samara University, Samara
e-mail: arina.lepalovskaya@mail.ru

Abstract

The paper addresses the problem of optimizing the computational complexity of the Iterative Soft Thresholding Algorithm (ISTA) for image reconstruction. The primary focus is on the transition from an interpreted Python implementation to compiled code using the Numba library. The effectiveness of JIT compilation and parallel computing is experimentally proven, resulting in a significant reduction in data processing time while maintaining high reconstruction quality.

Keywords: sparse representation, ISTA, Numba, JIT compilation, code optimization, parallel computing.