МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторная работа №2
по курсу «Программирование графических процессоров»

Классификация и кластеризация изображений на GPU.

Выполнил: Корнев Максим Сергеевич
Группа: М8О-412Б-22
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, 2025

Условие

1. **Цель работы**. Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти и одномерной сетки потоков.
2. **Вариант 2.** Метод расстояния Махаланобиса.
   На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, число nc - количество классов. Далее идут nc строчек описывающих каждый класс. В начале j-ой строки задается число np_j -количество пикселей в выборке, за ним следуют np_j пар чисел - координаты пикселей выборки. $nc \leq 32$, $np_j \leq 2^{19}$, $w*h \leq 4 * 10^8$.

Пример:

Входной файл	hex: in.data	hex: out.data
in.data out.data 2 4 1 2 1 0 2 2 2 1 4 0 0 0 1 1 1 2 0	03000000 03000000 A2DF4C00 F7C9FE00 9ED84500 B4E85300 99D14D00 92DD5600 A9E04C00 F7D1FA00 D4D0E900	03000000 03000000 A2DF4C01 F7C9FE00 9ED84501 B4E85301 99D14D01 92DD5600 A9E04C01 F7D1FA00 D4D0E900
in.data out.data 5 4 5 0 0 2 6 1 1 1 6 2 0 7 1 1 0 1 2 6 0 4 0 4 3 0 3 1 0 1 0 0 4 0 3 6 2 5 2 7 2 9 6 4 5 1 7 0 2 1 2 3 4 1 1 5 3 3 2 6	03000000 03000000 A2DF4C00 F7C9FE00 9ED84500 B4E85300 99D14D00 92DD5600 A9E04C00 F7D1FA00 D4D0E900	08000000 08000000 D2E27502 CFF65201 D3ED5701 D6E76902 C8F35B01 8E168200 CFF45001 AE977604 D3DC7102 7D1E7B00 AB9A8004 D9E58602 AB967E04 AE9D8004 87058200 D0F95B01 74148000 D0F55901 86136C00 85077400 D6E27702 D3609F03 D1609F03 CC5EA103 CC739D03 7C127F00 AA988804 AFA07D04 D0E37702 7D117A00 D6EB5901 D6E37C02 C9F85701 D655A103 D7EA7402 93127D00 D35BA403 D4DD7902 B0A18404 D6DE7502

Программное и аппаратное обеспечение

Аппаратное обеспечение

Графический процессор (GPU):

- Модель: NVIDIA Tesla T4
- Архитектура: Turing
- Compute Capability: **7.5**
- Графическая память: **16 ГБ GDDR6**
- Пропускная способность памяти: ~320 ГБ/с
- Количество мультипроцессоров (SM): **40**
- Максимальное число потоков на один SM: 1024
- Максимальное число потоков в блоке: 1024
- Максимальное число регистров на блок: 65 536 (по 32 бита)
- Разделяемая память на блок: до **64 КБ**
- Константная память: **64 КБ**

Процессор (CPU):

- Виртуальная машина Colab: 1 × Intel Xeon (Google Cloud)
- Частота: ~2.0–2.2 ГГц
- Количество ядер: обычно 2 доступных потока

Оперативная память (RAM):

- В Google Colab стандартно: **12–13 ГБ** (доступные пользователю)

Жёсткий диск (HDD/SSD):

- Виртуальный диск Colab: ~70–80 ГБ (SSD Google Cloud)

Программное обеспечение

- Операционная система: **Linux (Ubuntu 20.04 LTS, Google Colab среда)**
- Драйвер CUDA: **550.54.15**
- CUDA Toolkit: **12.5** (с nvcc 12.5.82, поддержка до Compute Capability 9.0)
- Язык программирования: **C++ (CUDA C)**
- Компилятор: nvcc (NVIDIA CUDA Compiler Driver)
- IDE / среда разработки: **Google Colaboratory (Jupyter Notebook web-IDE)**
- Дополнительно: доступ к стандартным библиотекам C (cstdio, cstdlib, cmath), а также к CUDA Runtime API

Метод решения

Программа выполняет классификацию пикселей изображения методом расстояния Махаланобиса классификатора с использованием вычислений на GPU. На первом этапе из стандартного ввода считываются пути к входному и выходному файлам, затем загружается изображение в бинарном формате. После этого считывается количество классов и координаты обучающих точек для каждого класса. На CPU вычисляются статистические параметры: средние значения RGB-компонент и ковариационные матрицы для каждого класса, после чего ковариационные матрицы инвертируются. Полученные средние значения и обратные ковариационные матрицы копируются в константную память GPU для быстрого доступа. Данные изображения передаются в глобальную память видеокарты, после чего запускается kernel-функция с одномерной сеткой потоков, где каждый поток обрабатывает несколько пикселей по схеме Grid-Stride Loop. Для каждого пикселя вычисляется расстояние Махаланобиса до всех классов путем умножения вектора разности на обратную ковариационную матрицу, выбирается класс с максимальным значением дискриминантной функции, и номер класса записывается в альфа-канал пикселя. После завершения вычислений классифицированное изображение копируется обратно в память CPU и сохраняется в выходной файл с последующим освобождением GPU-ресурсов.

Описание программы

Программа представлена одним файлом lab3.cu, в котором реализована полная логика классификации пикселей изображения методом расстояния Махаланобиса на GPU с использованием технологии CUDA. Основным типом данных служит структура uchar4, описывающая пиксель с четырьмя 8-битными компонентами (R, G, B, A), где альфа-канал используется для хранения номера класса. Для контроля корректности

выполнения функций CUDA используется макрос CUDA_CHECK, который проверяет возвращаемый код ошибок и при необходимости выводит сообщение с указанием места сбоя и завершает программу. Дополнительно определена структура Pixel для хранения координат обучающих точек.

В функции main() осуществляется чтение путей к входному и выходному файлам из стандартного ввода. Далее открывается входной бинарный файл, из которого считываются ширина и высота изображения, а затем массив пикселей загружается в вектор std::vector<uchar4>. После загрузки изображения из стандартного ввода считывается количество классов и для каждого класса — список координат обучающих точек, которые сохраняются в двумерном векторе class_pixels.

На этапе обучения классификатора для каждого класса на CPU вычисляются статистические параметры: средние значения RGB-компонент путём суммирования значений обучающих пикселей и деления на их количество. Затем вычисляются ковариационные матрицы размером 3×3, где каждый элемент определяется как сумма произведений отклонений соответствующих компонент от средних значений, делённая на количество точек минус один. Полученные ковариационные матрицы инвертируются с помощью функции invert_matrix, которая использует метод алгебраических дополнений для вычисления обратной матрицы через определитель. Средние значения и обратные ковариационные матрицы копируются в константную память GPU через cudaMemcpyToSymbol для обеспечения быстрого доступа всем потокам.

Основные вычисления выполняются в ядре classify_kernel, которое запускается на GPU в виде одномерной сетки потоков размером 256 блоков по 256 потоков. Каждый поток обрабатывает несколько пикселей по схеме Grid-Stride Loop, что обеспечивает эффективное покрытие всего изображения независимо от его размера. Для каждого пикселя вычисляется вектор разности между его RGB-компонентами и средними значениями каждого класса. Затем выполняется умножение вектора разности на обратную ковариационную матрицу с получением промежуточного вектора, после чего вычисляется скалярное произведение промежуточного вектора и вектора разности, что даёт квадрат расстояния Махаланобиса с обратным знаком. Среди всех классов выбирается тот, для которого дискриминантная функция (отрицательное квадратичное расстояние) максимальна, и его номер записывается в альфа-канал пикселя.

После завершения обработки данные копируются обратно в оперативную память с помощью cudaMemcpy, формируется бинарный выходной файл, содержащий ширину, высоту и массив классифицированных пикселей. Все выделенные ресурсы видеокарты, включая буфер изображения, корректно освобождаются перед завершением программы с помощью cudaFree.

Результаты

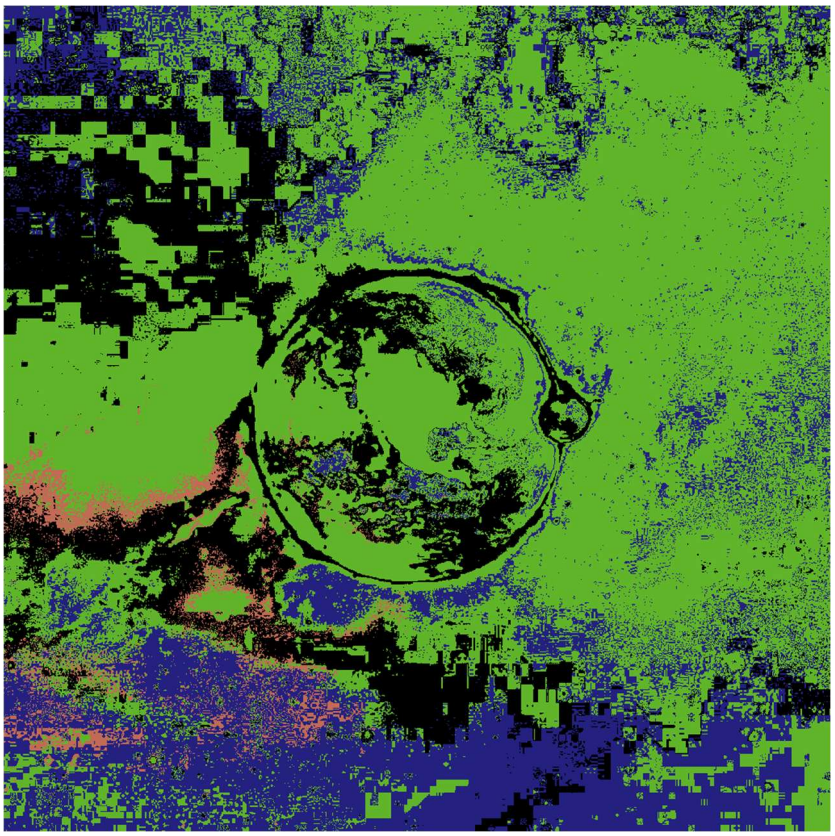
1. Сравнение результатов работы программы для различных конфигураций. Результаты отображены в таблице 1.

Конфигурация blocks, threads	**2 класса**	**3 класса**	**4 классов**
<<<1, 32>>>	67.396126	97.191612	117.396446
<<<32, 32>>>	2.113824	3.045856	3.669440
<<<64, 64>>>	1.208288	1.708704	2.285504
<<<128, 128>>>	1.204480	1.696000	2.306240
<<<256, 256>>>	1.052992	1.484768	1.992640
<<<512, 512>>>	**0.977760**	**1.378848**	**1.848096**
<<<1024, 1024>>>	0.974400	1.380576	1.850176
CPU	78.321227	132.126421	210.314441

Таблица 1 – таблица времени работы с различными конфигурациями

2. Примеры создания контура программой на GPU.

1024 * 1024



Выводы

В ходе работы был проведён анализ производительности алгоритма классификации пикселей методом расстояния Махаланобиса при различных конфигурациях сетки потоков на GPU. По полученным результатам видно, что время выполнения существенно зависит от выбранных параметров blocks и threads, а также от количества классов. При минимальной конфигурации <<<1, 32>>> наблюдается наихудшая производительность — от 67,4 мс для 2 классов до 117,4 мс для 4 классов, что связано с недостаточной параллелизацией вычислений.

С увеличением количества потоков время обработки резко сокращается. Наилучшие результаты достигаются при конфигурации <<<512, 512>>>, где время составляет 0,978 мс для 2 классов, 1,379 мс для 3 классов и 1,848 мс для 4 классов. Дальнейшее увеличение до <<<1024, 1024>>> не даёт улучшения производительности из-за насыщения аппаратных ресурсов GPU.

Сравнение с CPU-реализацией показывает значительное ускорение: для 2 классов GPU работает в 80 раз быстрее (0,978 мс против 78,3 мс), для 3 классов — в 96 раз быстрее, для 4 классов — в 114 раз. Использование константной памяти для хранения средних значений и обратных ковариационных матриц обеспечило эффективный доступ к параметрам классов для всех потоков одновременно.