Лабораторные курса ТПГУ.

1. Написать код, реализующий последовательный алгоритм умножения матриц и параллельный алгоритм для вычислений на GPU. Сравнить время выполнения.
2. Изменить код ядра в программе, вычисляющей произведение матриц на GPU таким образом, чтобы генерировать ошибку обращения к памяти определенными нитями.
    a. Обработать ошибку при выполнении ядра.
    b. Провести отладку программы с помощью cuda-gdb и выявить некорректно выполняемые нити.
3. Эмулировать недостаток регистров (большой размер локальных переменных в ядре) и, используя метрики ncu, определить использование локальной памяти.
4. Реализовать транспонирование матрицы размерностью N*K без использования разделяемой памяти, с разделяемой памятью без разрешения конфликта банков и с разрешением конфликта банков. С помощью метрик ncu:
    a. Определить время выполнения соответствующих ядер на GPU.
    b. Для всех трёх случаев определить эффективность использования разделяемой памяти.
    c. Определить для всех трех случаев пропускную способность при загрузке из глобальной памяти и при сохранении в глобальной памяти.
5. Включите в компоновку исполняемого файла (компоновщик nvcc) файлы .ptx, основываясь на процедуре, представленной в Лекции 4.
6. Провести сравнительный анализ производительности программ, реализующих произведение матриц с использованием *CUDA Runtime API* и *CUDA Driver API.*[*]
7. Провести сравнительный анализ производительности программ, реализующих произведение матриц с использованием *PyCuda*, *Numba cuda* и *numpy.matmul*.
8. Провести сравнительный анализ производительности программ, реализующих произведение матриц с использованием библиотеки *cuBLAS* и интерфейса *wmma*, при различных типах данных.
9. Переписать код обучения нейросети для распознавания цифр, представленный в Лекции 8, на языке C/C++ (с использованием CUDA API).
10. Разработать простой фреймворк (Python или C/C++) для обучения нейронных сетей.