Trabajo practico N° 3

Masi Juan, Perez Damián, Vildozola Mariano
37981647, 35375255, 32254403
Martes 19 a 23 hs, 6

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

Resumen. Se realizo la experiencia de Mean Filter (Filtro medio), realizado en Colab. Utilizando la ejecucion de programa CPU-CPU y CPU-GPU. En base a esta experiencia se analizo y se obtuvo las desventajas, ventajas y metricas de la programacion secuencial contra la programacion paralela. Tambien se realizo otra experiencia usando Keras, MNIST y CatBoost para clasificación de dígitos del 0 al 9 escritos a mano para crear una Convolutional Neural Network (CNN) para crear un modelo que identifica y predice una imagen de un dígito que nunca ha visto con un 99% de exactitud a mano.

Palabras claves: Pyton, GPU, mean filter, Cuda, Colab, CatBoost, Kera, Machine learning, Gradient Boosted Decision Trees.

1 Introducción

Google Colaboratory, también llamado "Colab", permite ejecutar y programar en Python directamente en el navegador. Posee ciertas ventajas: no requiere configuración, se pueden corren en la nube, es posible elegir correr en una CPU o GPU de forma gratuita, y permite compartir contenido fácilmente.

CUDA es una plataforma de computación paralela general y un modelo de programación. Puede administrar GPU, recursos de hardware y utlizar GPU para realizar computación compleja.
El programa CUDA consta de dos partes: una es el código de host que se ejecuta en la CPU y otra es el código del dispositivo ejecutándose en GPU. El programa paralelo que se ejecuta en la GPU también se conoce como kernel.

Ejercicio 1:
Mean filtering: El filtrado medio es un método de filtrado lineal, que reemplaza el valor de gris de píxel del centro punto de la ventana por medio de un píxel de ventana suave.
Dada una imagen f(i,j), el procedimiento consiste en generar una nueva imagen g(i,j) cuya intensidad para cada píxel se obtiene promediando los valores de intensidad de los píxeles f(i,j) incluidos en un entorno de vecindad predefinido.

En pocas palabras, es tomar el promedio de los valores de píxeles en un área determinada en lugar del valor de píxeles original.

La máscara de filtro 3 * 3 que se usa comúnmente es:

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

El proceso de filtrado de una imagen M × N a través de un filtro promedio ponderado m × n se puede dar mediante la siguiente fórmula:

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)} \quad (3.5-1)$$

Ejericio 2:
Utilizando las herramientas de Keras y data MNIST(Modified National Institute of Standards and Technology database), que contiene un conjunto de entrenamiento de 60000 imágenes de dígitos manuscritos, se va a predecir una imagen de un digito, que nunca ha visto con un 99% de exactitud.

2 Desarrollo

Ejercicio 1:

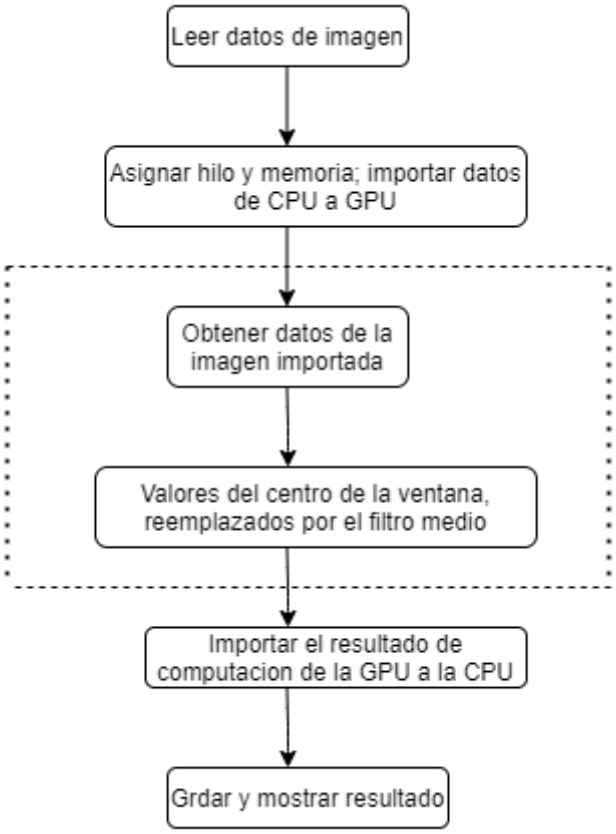
- Manual de usuario: Primero se debe ejecutar armado del ambiente y luego lectura de imagen de dominio público.

  Luego se tiene que ejecutar procesamiento y filtrado de imagen utilizando CPU. Con esta ejecución se va a visualizar la imagen original y la imagen aplicando Mean Filter, utilizando CPU.
  Después se tiene que ejecutar (Desarrollo GPU) Mean Filter utilizando CUDA. Con esta ejecución se va a visualizar la imagen original y la imagen aplicando Mean Filter, utilizando GPU.

- https://github.com/juanagustinmasi/VacunarTech/blob/main/TP3/Cuaderno_1_martes_grupo6_2021.ipynb

- El funcioamiento desde el punto de vista de la programacion en CPU es el siguiente: se realiza la apertura de la imagen, se recorre la misma, se le aplcia el filtro correspondiente y se muestra el resultado. Para realizar el recorrido de la imagen, se realiza de forma secuencial.
- El funciamiento desde el punto de vista de la programacion en GPU es el siguiente: se realiza la apertura de la imagen, luego se reserva en la memoria de la CPU y dicha porcion de memoria se copia a la memoria de la GPU. Luego de acuerdo a las dimensiones de la iamgen se recorre con los hilos los pixeles de la imagen, se aplica el filtro y se obtiene el resulado. Para realizar el recorrido de la imagen, se realiza con los hilos, ingresando a cada pixel de forma paralela.

- Diagrama de flujo del algoritmo de filtrado medio



Los pasos en la línea de puntos se ejecutan en el extremo del dispositivo (GPU) y otros pasos ejecutar en el extremo del host (CPU).

- Resultados de experimentos:

- Al planificar con la mitad de los hilos-gpu, aumenta un poco el tiempo de procesamiento de la imagen en comparacion a utlizando los hilos-gpu que se tenian.
- Al planificar con el dobel de hilos-gpu, el tiempo de procesamiento baja un poco. Pero no es un tiempo de gran importancia como para decidir utlizar el doble de hilos.
- Al planificar con la cantidad maxima de hilos por bloques soportadas por el GPGPU, se obtiene un menor tiempo que el original, pero tampoco varia mucho el tiempo de procesamiento.

Ejercicio 2:

- En este notebook estamos utilizando redes neuronales convulsiónales, es un tipo de red neuronal artificial con aprendizaje supervisado que procesa sus capas imitando la corteza visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que puedan identificar objetos y producir inferencias a partir de los datos disponibles, gracias a su capacidad de aprendizaje.

- Inputs: Se tienen las 60000 imágenes prevenientes de la base de datos de MNIST.

- Output: Imagen procesada inferida, resultado de haber ejecutado esta red neuronal.

- Manual de usuario: Primero se debe ejecutar el armado del ambiente, donde se almacenan las imágenes, recibidas del dataSet de MNIST.
  Como entorno de ejecución se tiene que cambiar el tipo de entrono de ejecución y como acelerador de hardware, elegir GPU.
  Luego se define el armado y ejecución del modelo, donde se definen los parámetros del modelo, en el cual hay un parámetro *task_type*, el cual sirve para seleccionar el entorno de ejecución, es decir CPU o GPU.
  Después de realizar esto se ejecuta visualizando la imagen del digito predecido.

- https://github.com/juanagustinmasi/VacunarTech/blob/main/TP3/Cuaderno_2_martes_grupo6_2021.ipynb

3 Conclusiones

Ejercicio 1:

Comparación del tiempo de ejecución de filtro medio de un programa por CPU y programa por CUDA.

	TIPO DE DEFECTO			
Programa	Punto negro	Inclusión	Raya	Punto blanco
Tiempo de ejecución del programa CPU (ms)	4.36	4.04	4.20	4.34
Tiempo de ejecución del programa CUDA (ms)	1.28	1.09	1.26	1.11
Relación de aceleración	3.41	3.71	3.33	3.91

Analizando los resultados de la tabla podemos conocer que el programa por CUDA ejecuta a nivel mas rapido que los programas por CPU.

Ejercicio 2:
Uno de los problemas surgidos en el desarrollo del ejercicio, fue utilizar el procesamiento con CPU ya que tardaba demasiado en ejecutar para obtener los resultados, lo que demoraba las pruebas para comparar la eficiencia entra GPU y CPU.
Adicionalmente, tuvimos que descargar la biblioteca CatBoost porque no esta incluida en la API de tensorFlow.

La conclusion de mayor importancia que encontramos, realizando este notebook, es la gran diferencia de tiempo que hay utlizando el procesamiento con GPU contra el procesamiento con CPU.

4 Referencias

1- https://towardsdatascience.com/image-filters-in-python-26ee938e57d2

2- https://www.programmersought.com/article/49584514760/

3- https://www.programmersought.com/article/72128196687/

4- http://www.librow.com/articles/article-5

5- https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ipr.2013.0521

6- https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ipr.2017.0825

7- https://keras.io/

8- https://catboost.ai/

9- https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119564843.ch5

10- https://www.sciencedirect.com/science/article/abs/pii/S0040162521000901