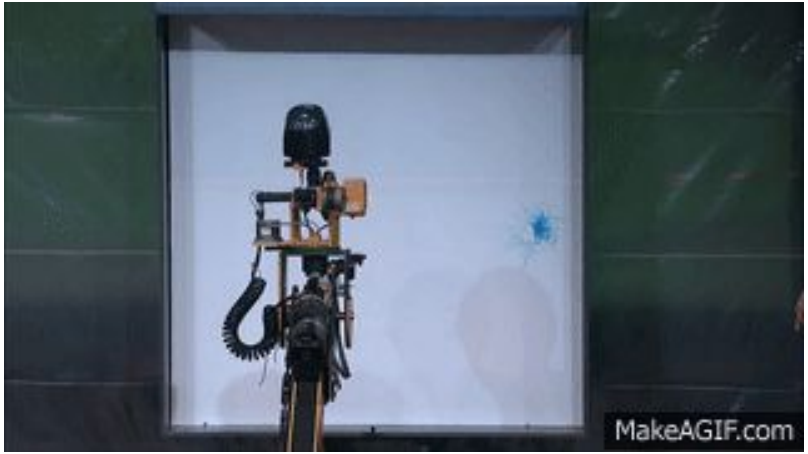
CUDA

By: Guillermo Andres De Mendoza Corrales

CPU vs GPU



- Central processing unit
- **Versatil**
- 1 to 30 cores
- Low latency GHz
- Serial

- Graph processing unit
- **Only some task (vectors)**
- 100s to 10000s cores
- High throughput (MHz)
- Parallel

Architecture comparison

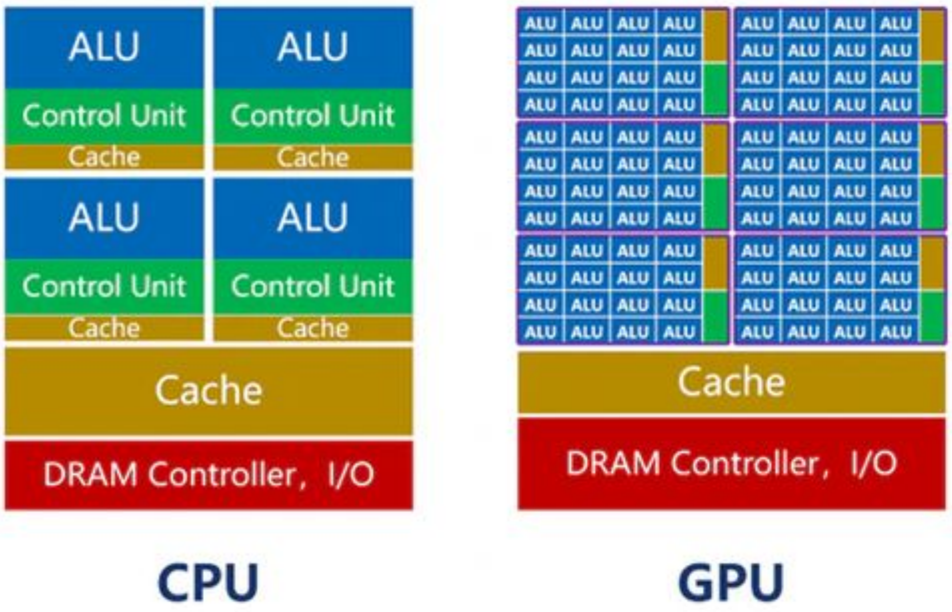
ALU: Arithmetic Logic Unit

Control Unit:

- Search instructions
- Decode instructions
- Control executions
- Manage flows *by buss
- Synchronize

Cache:

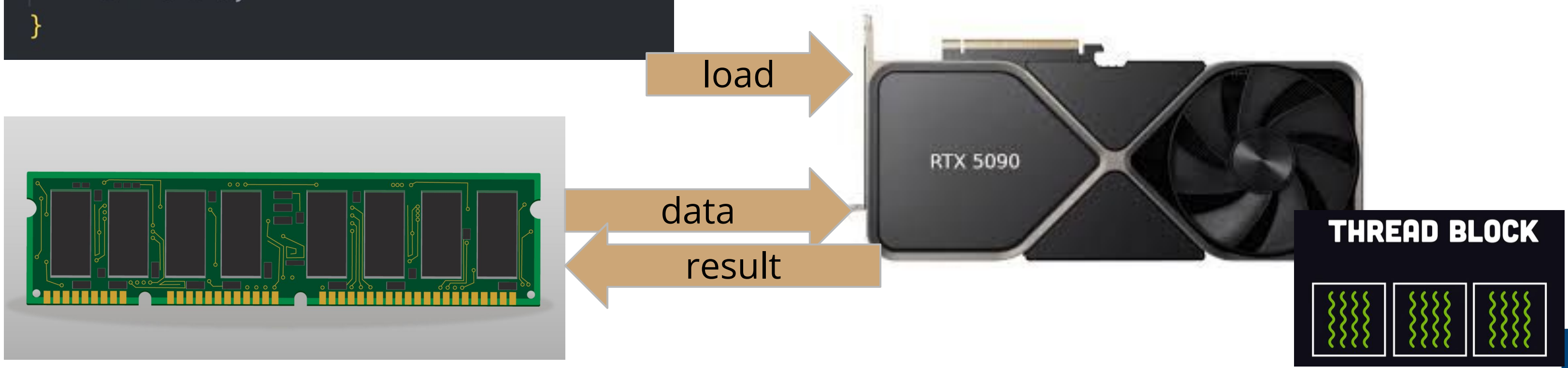
- Contain instructions
- Data access
- L1 -> L2 -> L3 -> RAM



CUDA

CUDA (Compute Unified Device Architecture) is an architecture for parallel computing, developed by **NVIDIA in 2007**, that leverages the power of its GPUs for general-purpose processing (not only graphical). It uses a hierarchy of grids, thread blocks, and threads to execute tasks in parallel, significantly speeding up compute-intensive application.

```
__global__ void add(int a, int b, int *c) {
    *c = a + b;
}
```



CC : Compute Capability

Define the allowed functions and characteristics that the GPU can perform.

Each CUDA version require a minimum CC value

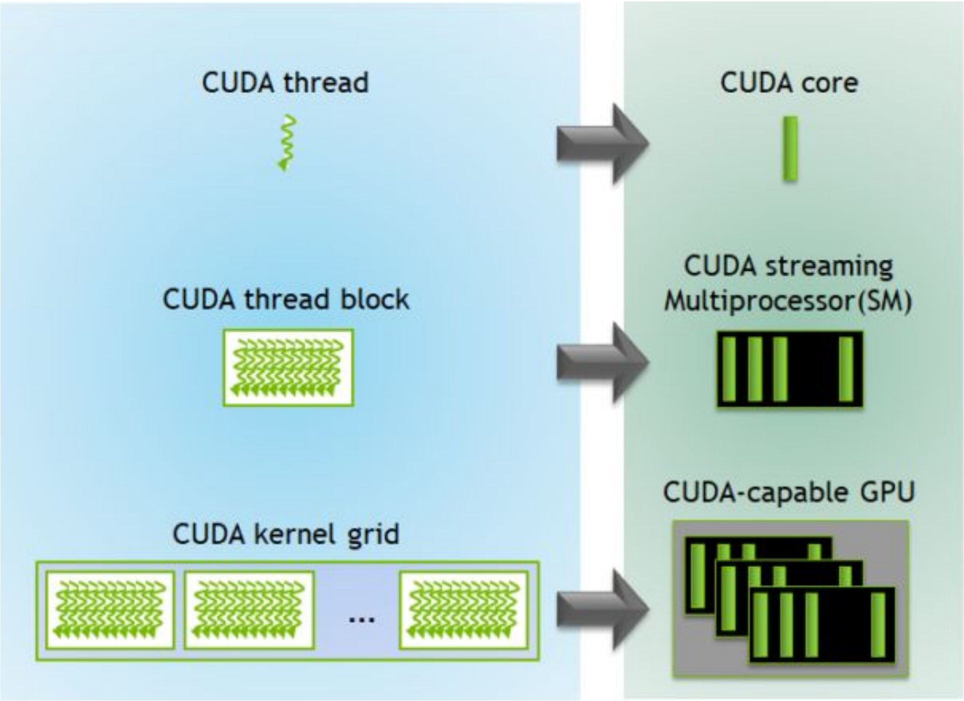
Some advance functions require also a minimum CC value

- Tensor cores
- Ray tracing
- Acceleration FP16
- Some libraries

Capacidad de cómputo	Centro de datos	Estación de trabajo/Consumidor	Supersónico
12.0	NVIDIA RTX PRO 6000 Blackwell Server Edition	NVIDIA RTX PRO 6000 Blackwell Edición para estación de trabajo NVIDIA RTX PRO 6000 Blackwell Edición para estación de trabajo Max-Q NVIDIA RTX PRO 5000 Blackwell NVIDIA RTX PRO 4500 Blackwell NVIDIA RTX PRO 4000 Blackwell NVIDIA RTX PRO 4000 Blackwell Edición SFF NVIDIA RTX PRO 2000 Blackwell GeForce RTX 5090 GeForce RTX 5080 GeForce RTX 5070 Ti GeForce RTX 5070 GeForce RTX 5060 Ti GeForce RTX 5060 GeForce RTX 5050	
11.0			Jetson T5000 Jetson T4000
10.3	NVIDIA GB300 NVIDIA B300		
10.0	NVIDIA GB200 NVIDIA B200		
9.0	NVIDIA GH200 NVIDIA H200 NVIDIA H100		
8.9	NVIDIA L4 NVIDIA L40 NVIDIA L40S	NVIDIA RTX 6000 Ada NVIDIA RTX 5000 Ada NVIDIA RTX 4500 Ada NVIDIA RTX 4000 Ada NVIDIA RTX 4000 SFF Ada NVIDIA RTX 2000 Ada GeForce RTX 4090 GeForce RTX 4080 GeForce RTX 4070 Ti GeForce RTX 4070 GeForce RTX 4060 Ti GeForce RTX 4060 GeForce RTX 4050	
8.7			Jetson AGX Orin Jetson Orin NX Jetson Orin Nano
8.6	NVIDIA A40 NVIDIA A10 NVIDIA A16 NVIDIA A2	NVIDIA RTX A6000 NVIDIA RTX A5000 NVIDIA RTX A4000 NVIDIA RTX A3000 NVIDIA RTX A2000 GeForce RTX 3090 Ti GeForce RTX 3090 GeForce RTX 3080 Ti GeForce RTX 3080 GeForce RTX 3070 Ti GeForce RTX 3070 GeForce RTX 3060 Ti GeForce RTX 3060 GeForce RTX 3050 Ti	

KERNEL

Each thread execute a copy of the kernel



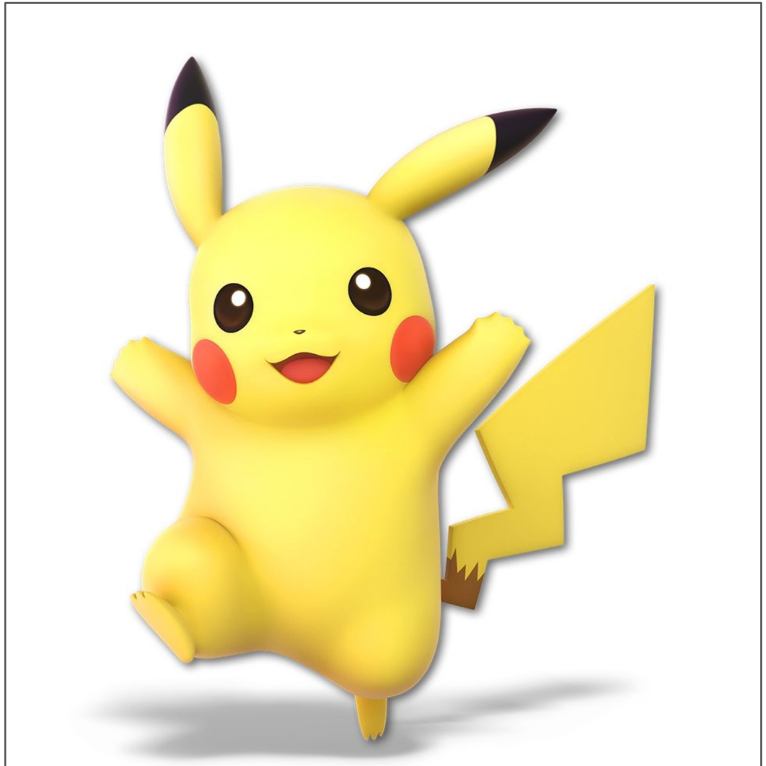
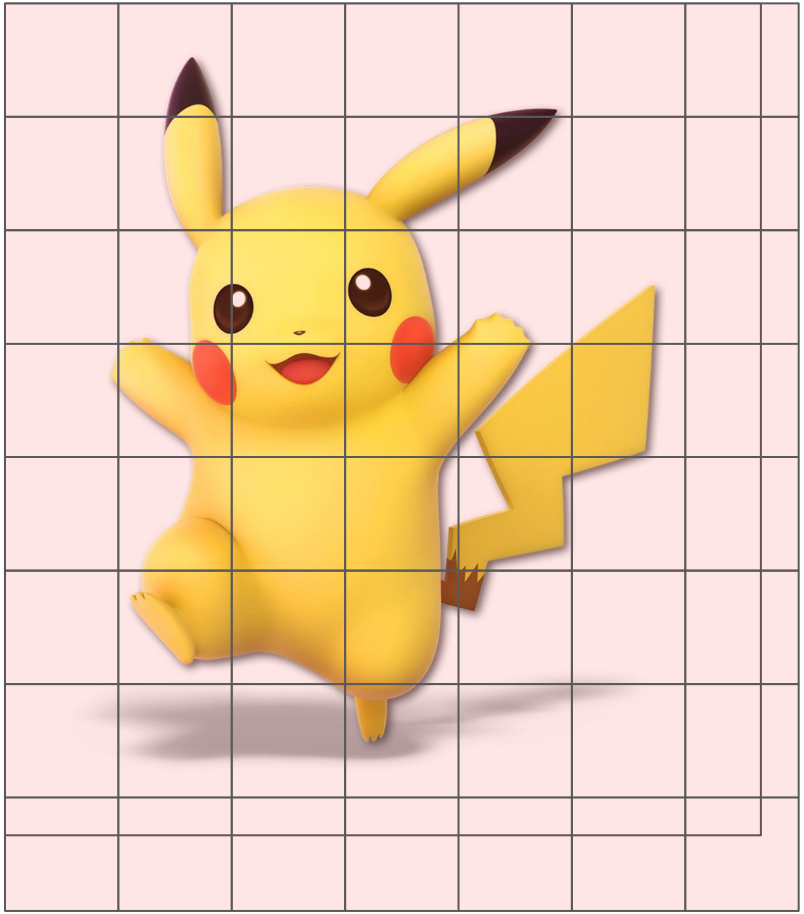


imagen: y:110 , x:100

ThreadBlock: 15x15

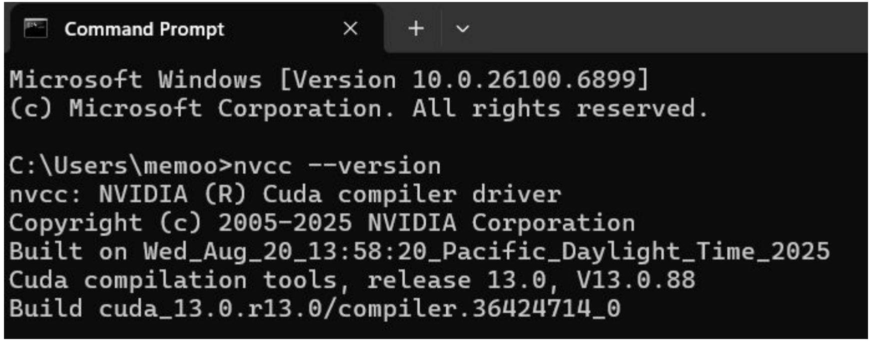
Grid: (x:7, y:8)



KERNEL

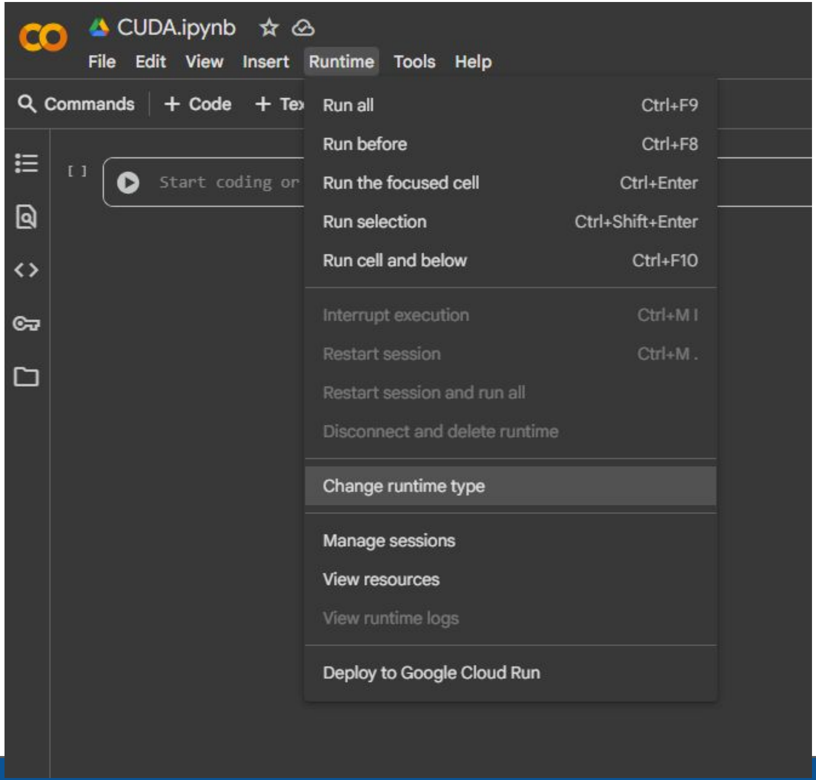
1. **CUDA Toolkit**
- CUDA compiler nvcc
- CUDA Libraries
- CUDA Runtime
- CUDA Driver APIs
- CUDA Debugging
- CUDA Performance Analysis

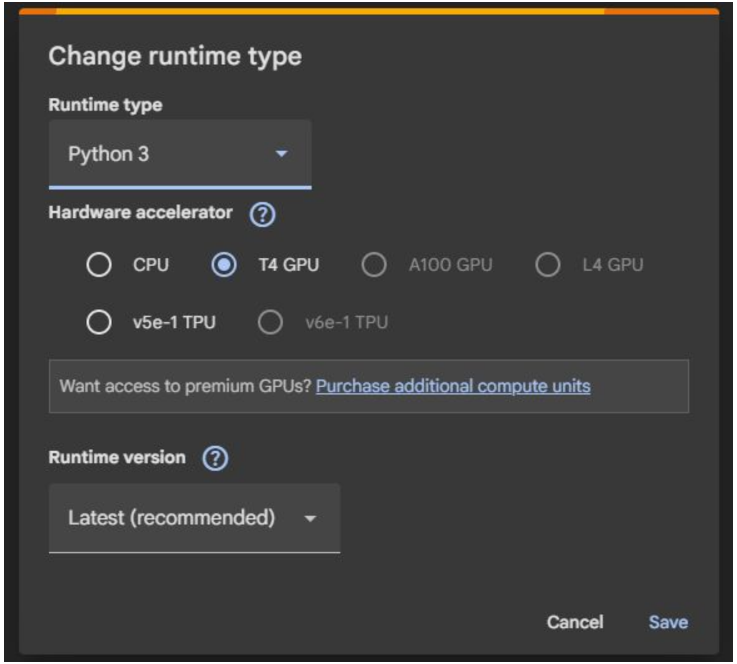
1. verify CUDA version
- nvcc --version



COLAB IMPLEMENTATION

Set notebok to GPU





Hardware enviorment

```
[1]
4s
from numba import cuda
print(cuda.is_available())
cuda.detect()
```

```
True
Found 1 CUDA devices
id 0             b'Tesla T4'                              [SUPPORTED]
                      Compute Capability: 7.5
                           PCI Device ID: 4
                              PCI Bus ID: 0
                                    UUID: GPU-4637cb14-ea62-ed4a-90d3-2709af8ff9c3
                                Watchdog: Disabled
             FP32/FP64 Performance Ratio: 32
Summary:
        1/1 devices are supported
True
```

GPU Architecture	**NVIDIA Turing**
NVIDIA Turing Tensor Cores	**320**
NVIDIA CUDA® Cores	**2,560**
Single-Precision	**8.1 TFLOPS**
Mixed-Precision (FP16/FP32)	**65 TFLOPS**
INT8	**130 TOPS**
INT4	**260 TOPS**
GPU Memory	**16 GB GDDR6 300 GB/sec**
ECC	**Yes**
Interconnect Bandwidth	**32 GB/sec**
System Interface	**x16 PCIe Gen3**
Form Factor	**Low-Profile PCIe**
Thermal Solution	**Passive**
Compute APIs	**CUDA, NVIDIA TensorRT™, ONNX**

Hardware enviorment

```
!nvidia-smi

Sat Oct 18 02:49:58 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   56C    P8             10W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

CODE

Avoid errors

```
from google.colab import files
import cv2
import os
import shutil
import matplotlib.pyplot as plt
```

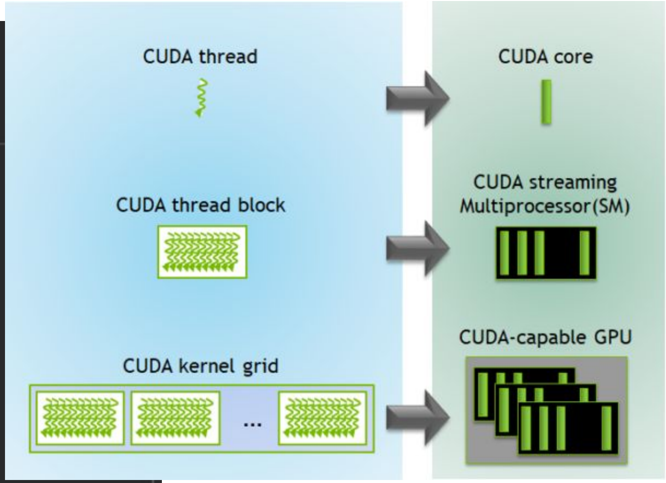
```
import numpy as np
from numba import cuda
from numba import config
config.CUDA_ENABLE_PYNVJITLINK = 1 # resuelve bug de versiones
```

Kernel

```
@cuda.jit
def grayscale_kernel_2D(img_color, img_gray):
    """
    Kernel 2D que mapea las coordenadas (i, j) directamente usando
    índices de bloque y de hilo en las dimensiones X e Y.
    i = fila (altura), j = columna (ancho)
    """
    # Obtener las coordenadas 2D (i, j)
    # cuda.grid(2) devuelve (j, i) -> (column,row)
    j, i = cuda.grid(2)

    # Obtener las dimensiones de la imagen de salida (filas, columnas)
    rows, cols = img_gray.shape

    # Chequeo de límites
    # Asegurarse de que las coordenadas estén dentro de los límites de la imagen
    if i < rows and j < cols:
        # Acceder a los componentes de color de la imagen original
        R = img_color[i, j, 0]
        G = img_color[i, j, 1]
        B = img_color[i, j, 2]

        # Cálculo de la intensidad de gris
        # Se mantienen los coeficientes estándar
        gray_value = 0.2989 * R + 0.5870 * G + 0.1140 * B

        # Asignar el valor de gris al array de salida
        img_gray[i, j] = gray_value
```

GPU Caller

```
def image_to_grayscale(img):

    # 'img' entra como uint8 (0-255)
    ROWS, COLUMNS, COLOR = img.shape

    # Transferir la imagen original a la GPU
    d_img_color = cuda.to_device(img)

    # Reservar memoria en GPU para la imagen de salida
    d_img_gray = cuda.device_array((ROWS, COLUMNS), dtype=np.float32)

    # Configuración del GRID
    threadsperblock = (16, 16)
    blockspergrid_x = (COLUMNS + threadsperblock[0] - 1) // threadsperblock[0]
    blockspergrid_y = (ROWS + threadsperblock[1] - 1) // threadsperblock[1]
    blockspergrid = (blockspergrid_x, blockspergrid_y)

    # Lanzamiento
    grayscale_kernel_2D[blockspergrid, threadsperblock](d_img_color, d_img_gray)

    # Recuperar (copy_to_host ya sincroniza implícitamente)
    return d_img_gray.copy_to_host()
```



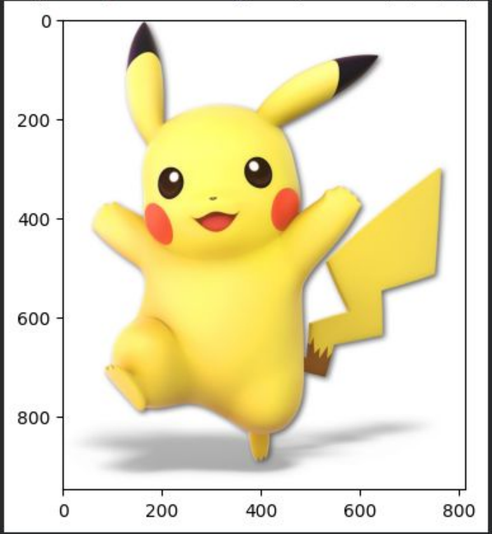
Read Image

```
uploaded = files.upload()

img_color_name = list(uploaded.keys())[0]
img_color = cv2.imread(img_color_name)
img_color_cv_rgb = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)
img_color_np = img_color_cv_rgb.astype(np.float32) / 255.0
H, W, C = img_color_np.shape
print(f"Imagen cargada como NumPy array. Forma (H, W, C): {H}x{W}x{C}")

plt.imshow(img_color_np)
plt.show()
```

Elegir archivos pikachu.png

pikachu.png(image/png) - 462351 bytes, last modified: 5/2/2026 - 100% done

```
Saving pikachu.png to pikachu.png
Imagen cargada como NumPy array. Forma (H, W, C): 948x812x3
```



Logic

```
img_gray = image_to_grayscale(img_color_np)
```

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
# img_np debe normalizarse si no está en el rango [0, 255]
plt.imshow(img_color_np / 255 if img_color_np.max() > 1 else img_color_np)
plt.title("Imagen Original")
plt.axis('off')

plt.subplot(1, 2, 2)
# El resultado es float32, usamos cmap='gray'
plt.imshow(img_gray, cmap='gray')
plt.title("Imagen en Blanco y Negro (CUDA 2D)")
plt.axis('off')

plt.show()
```