```
from numba.cuda.cudadrv import enums
from numba import cuda
```

```
!wget -O img.jpeg https://file.io.vn/1H0WQ5
```

```
--2025-10-03 04:06:19--  https://file.io.vn/1H0WQ5
Resolving file.io.vn (file.io.vn)... 157.10.45.101
Connecting to file.io.vn (file.io.vn)|157.10.45.101|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://my.microsoftpersonalcontent.com/personal/de4e2fbd58f4f53b/_layouts/15/download.aspx?UniqueId=26cb7
--2025-10-03 04:06:24--  https://my.microsoftpersonalcontent.com/personal/de4e2fbd58f4f53b/_layouts/15/download.aspx
Resolving my.microsoftpersonalcontent.com (my.microsoftpersonalcontent.com)... 13.107.137.11, 13.107.139.11
Connecting to my.microsoftpersonalcontent.com (my.microsoftpersonalcontent.com)|13.107.137.11|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 75109 (73K) [image/jpeg]
Saving to: 'img.jpeg'

img.jpeg            100%[===================>]  73.35K   329KB/s    in 0.2s

2025-10-03 04:06:25 (329 KB/s) - 'img.jpeg' saved [75109/75109]
```

```
# from matplotlib import pyplot
import matplotlib
```

```
#Load the image
img = matplotlib.pyplot.imread('img.jpeg')

matplotlib.pyplot.imshow(img)
```

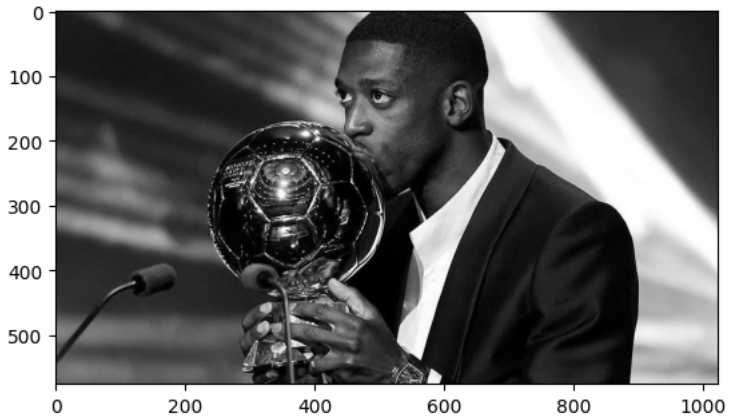
```
<matplotlib.image.AxesImage at 0x7ae5d8577170>
```


```
height, width, channels = img.shape
rgb_1d = img.reshape(height * width * 3)
```

```
import time
```

```
# CPU
start_time = time.time()
rgb_2d = rgb_1d.reshape(height, width, 3)
gray_2d = 0.299 * rgb_2d[:, :, 0] + 0.587 * rgb_2d[:, :, 1] + 0.114 * rgb_2d[:, :, 2]
end_time = time.time()
print("CPU time: ", end_time - start_time)
matplotlib.pyplot.imshow(gray_2d, cmap='gray')
```

```
CPU time:  0.010538578033447266
<matplotlib.image.AxesImage at 0x7ae5c781aa20>
```

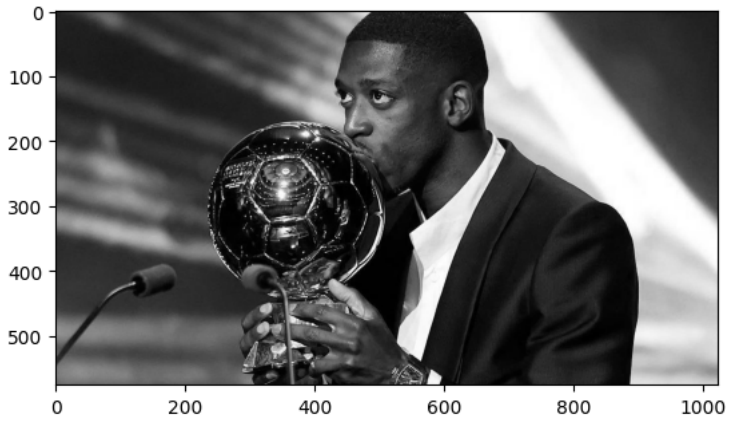


```
from numba import jit, int32
import numpy as np
from numba import config
import matplotlib.pyplot as plt
config.CUDA_ENABLE_PYNVJITLINK = 1
```

```
@cuda.jit
def grayscale(src, dst):
  tidx = cuda.threadIdx.x + cuda.blockIdx.x * cuda.blockDim.x
  pixel_idx = tidx * 3
  if pixel_idx + 2 < src.shape[0]:
    g = np.uint8((src[pixel_idx] + src[pixel_idx + 1] + src[pixel_idx + 2]) / 3)
    dst[pixel_idx] = g
    dst[pixel_idx + 1] = g
    dst[pixel_idx + 2] = g
```

```
blockSize = 32
pixelCount = height * width
gridSize = (pixelCount + blockSize - 1) // blockSize

devSrc = cuda.to_device(rgb_1d)
devDst = cuda.device_array(height * width * 3, np.uint8)
start_time_gpu = time.time()
grayscale[gridSize, blockSize](devSrc, devDst)
end_time_gpu = time.time()
hostDst = devDst.copy_to_host()

print("GPU time: ", end_time_gpu - start_time_gpu)
matplotlib.pyplot.imshow(hostDst.reshape(height, width, 3), cmap='gray')
```

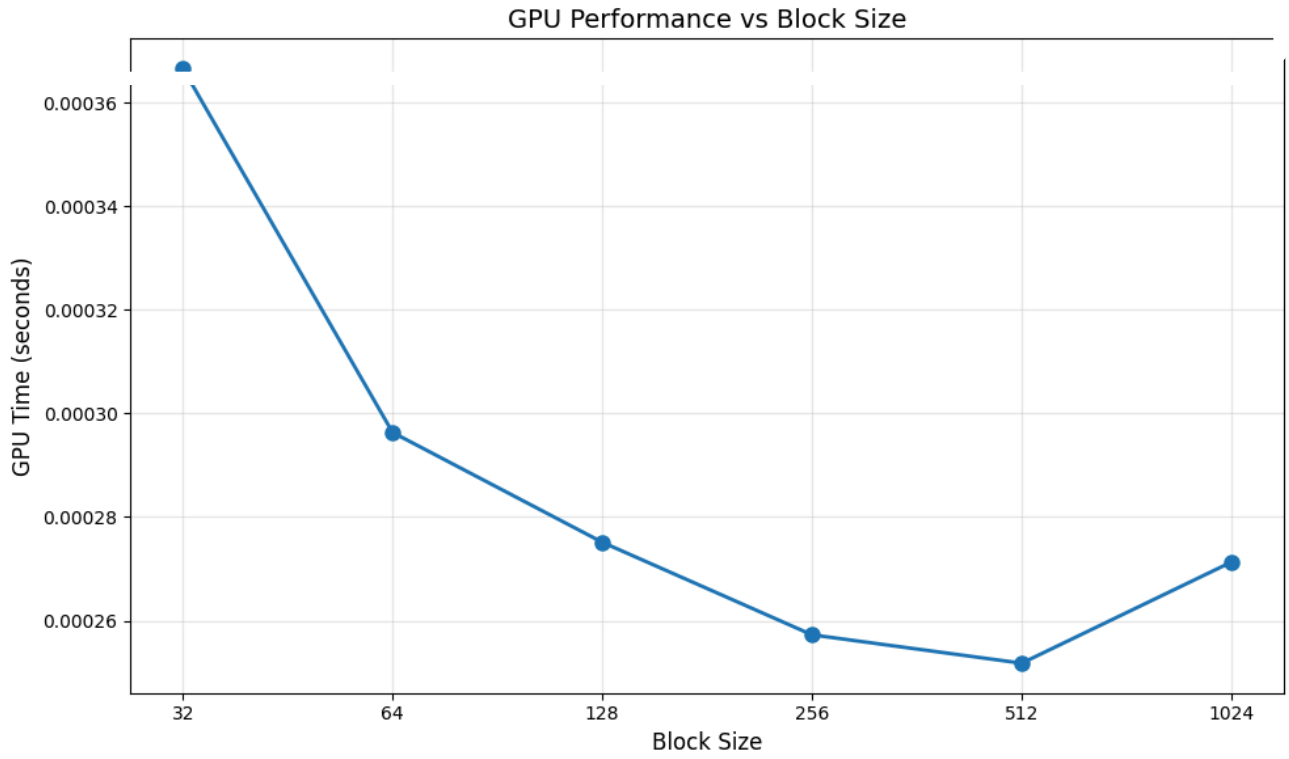
```
GPU time:  1.349064826965332
<matplotlib.image.AxesImage at 0x7ae59db3e690>
```



```
block_sizes = [32, 64, 128, 256, 512, 1024]
gpu_times = []

pixelCount = height * width

for blockSize in block_sizes:
    gridSize = (pixelCount + blockSize - 1) // blockSize

    devSrc = cuda.to_device(rgb_1d)
```

```
    devDst = cuda.device_array(height * width * 3, np.uint8)

    start_time_gpu = time.time()
    grayscale[gridSize, blockSize](devSrc, devDst)
    cuda.synchronize()
    end_time_gpu = time.time()

    gpu_time = end_time_gpu - start_time_gpu
    gpu_times.append(gpu_time)

    print(f"Block Size: {blockSize}, GPU time: {gpu_time} seconds")

plt.figure(figsize=(10, 6))
plt.plot(block_sizes, gpu_times, marker='o', linewidth=2, markersize=8)
plt.xlabel('Block Size', fontsize=12)
plt.ylabel('GPU Time (seconds)', fontsize=12)
plt.title('GPU Performance vs Block Size', fontsize=14)
plt.grid(True, alpha=0.3)
plt.xscale('log', base=2)
plt.xticks(block_sizes, block_sizes)
plt.tight_layout()
plt.show()
```

```
Block Size: 32, GPU time: 0.0003666877746582031 seconds
Block Size: 64, GPU time: 0.0002963542938232422 seconds
Block Size: 128, GPU time: 0.0002751350402832031 seconds
Block Size: 256, GPU time: 0.00025725364685058594 seconds
Block Size: 512, GPU time: 0.00025177001953125 seconds
Block Size: 1024, GPU time: 0.0002713203430175781 seconds
```




```
best_idx = min(gpu_times)
print(best_idx)
```

```
0.0002467632293701172
```