In homework number two the goal was to improve the speed of the serial-programmed application by using multithreading and multiprocessing. We use threading when it is I/O bound operation and multiprocessing when it is a CPU bound operation.

In the assignment there were two tasks, the first task was to gather images from the Imgur api and save them in a folder. This task is a I/O bound operation, which means threading should be used to improve the speed. The second task was to resize the images, which is a CPU bound operation. Since it is a CPU bound operation multiprocessing should be used.

In task one, gathering the images from Imgur, the library “concurrent.future” was used. The function ThreadPoolExecutor with ten max workers was used. ThreadPoolExecutor uses a pool of threads to execute calls in a concurrent fashion. As seen in the code in ‘Assignment1.ipynb’ the serial-programming took 1 min 37s and multithreading took only 33.5 s to do the same task.

In task two, resizing the images, the library “multiprocessing” was used. The ‘Pool’ function was used with a pool size of 3, which means that three executions of the function are happening in parallel. As seen in the code in ‘Assignment1.ipynb’ the serial-programming took 5.71 s and multiprocessing took only 32.6 ms to do the same task.

In both instances using multithreading or multiprocessing showed an improvement in the speed from the serial-programmed application. Understanding if it is a I/O bound operation or CPU bound operation is the key to choosing which technique is needed to speed up the serial-programmed application.