This mini-project is able to search through a Twitter feed, download a certain number of images, get a description of the images, and compile the images into a video. The code looks very good; there is a description for every function, and everything is done efficiently. Error handling could be improved by handling each section individually and providing specific error messages to the user. The runtime of the code depends on the size of the Twitter feed, the size of the images, and Google Vision. The most significant bottleneck by far is the Vision API, which can take over five seconds per image to analyze and provide a description. CPU and memory usage is minimum from the library itself, but ffmpeg uses a significant amount of resources depending on the quality and quantity specified in settings. The mini-project is synchronous because the user is unable to run code while images are downloaded and analyzed. A better approach is to use an asynchronous runtime, especially since the process can take a while to complete.

While this is an excellent library, there could be some improvements. The description of images is never returned to the user; instead, it is directly written on the video. This poses a problem if the user needs to use the description separate from the video. Furthermore, the images are also deleted after downloaded, requiring the user to modify the library code to prevent this from occurring. Asynchronous operation would greatly improve usability and allow the user to perform operations during image download/processing.

Performance results:

Test1: 2 images

Runtime: 9.06374756799778 seconds

Memory Peak: 8800610 bytes

Test2: 4 images

Runtime: 14.99528017299599 seconds

Memory Peak: 8767619 bytes

Test3: 2 images, error

Runtime: n/a

Memory Peak: n/a