Scientific Computation
Autumn 2023
Project 4

Due: Monday January 15th 1:00pm

In addition to this file, there are three files for this assignment: 1) *project4.py*, a Python module which you will complete and submit on Blackboard (see below for details), 2) *report4.tex*, a template file for your short report which will also be submitted on Blackboard, and 3) *project4.ipynb*, a Jupyter notebook which you will complete and submit.

If you have not done so already, read sections 1-3 and section 4.1 of: Kisil, I., Calvi, G. G., Scalzo Dees, B., & Mandic, D. P. (2020). Tensor Decompositions and Practical Applications: A Hands-on Tutorial. This paper is referred to as KCSM below.

Part 1

1. (10 pts) Design a Jupyter notebook that guides a Scientific Computation student that has not read KCSM through the process of compressing and then reconstructing a color image using the tensor-train decomposition. The decomposition should follow algorithm 3 in KCSM. After completing the notebook, the student should be able to: 1) explain the tensor operations needed to carry out the computations, 2) implement the tensor-train decomposition and reconstruction, and 3) draw one or more substantive connections between this image processing task and content covered during the term. You should design a sequence of tasks along with explanations providing essential background information and motivation for the tasks that help the student reach these three objectives. The notebook should include a clear explanation of line 8 in algorithm 3 (what is its underlying logic? why is it correct?). Your work will be assessed based on its correctness, originality, and the degree to which it illustrates a good understanding of the relevant material from KCSM and the module. It is fine for the design of the notebook to be 'basic' (like our lab notebooks), and there is no need for elaborate pictures or diagrams. A student should be able to complete the given tasks in roughly 1 hour or less. Code has been provided to you in *project4.ipynb* to load a color image. You may use numpy, matplotlib, and scipy as needed. Please do not use other packages without permission. The notebook you submit should contain both the instructions for the hypothetical student and the "solution" for the tasks you have designed. Use the solutions for your notebook to complete the functions *decompose1* and *reconstruct* in *project4.py*.

   Note: If you use *np.linalg.svd*, it is recommended that you set *full_matrices=False*.

2. (2 pts) Algorithm 3 in KCSM requires an accuracy parameter to be specified. An alternative is to specify $N-1$ values for the rank. Complete *decompose2* to decom-

pose a tensor using $N - 1$ rank values specified as input rather than ϵ but which otherwise follows the same approach as *decompose1*. Critically compare these two approaches, and explain their relative strengths and weakness in your report.

Part 2

(8 pts) In this part, you will analyze and critically compare three approaches for tensor compression: TTSVD (using either of the approaches from part 1.2), HOSVD (following the approach described in KCSM), and repeated application of low-rank matrix approximations. Your analysis should be based on careful consideration of these algorithms as well as well-designed quantitative computational tests where the algorithms are applied to 1) the color image from part 1 and 2) a short color video of your choice. You should first provide an in-depth discussion of the relative strengths and weakness of the three methods when applied to the image. Then choose the two methods which you believe are best-suited for video compression, apply them to your video, and critically compare the two methods. For both the image and the video, analysis and interpretation of test results should be explicitly connected to your understanding of the algorithms. You may use the HOTTBOX package in this question, however it is recommended that you develop your own code for the HOSVD. You have been provided functions which will 1) convert a mp4 video file into a numpy array and 2) convert a numpy array (with appropriate shape) into a mp4 video file. Note that higher-resolution and longer-duration videos will require more memory. A video that lasts a few seconds and which has a moderately lower resolution than the color image is fine. Your discussion and supporting figures should be presented in your report. Place code for your tests, analysis, and figures in the *part2* function. Include your original mp4 file with your submission.

Note: The code for reading/writing mp4 files requires the opencv package which can be installed in Anaconda navigator or by using pip: pip3 install opencv-python

Further guidance

1. You should add code to the name==main portion of your Python file to call *part2* and generate the figure(s) included in your report.
2. You may add additional functions as needed.
3. This assignment is for year-4 (MSci) and MSc students.
4. I am happy to provide help with problems related to creating/editing Jupyter notebook cells for part 1 or finding an appropriate video for part 2.
5. Your submission should represent your own work unless stated otherwise.

Submitting the assignment

To submit the assignment for assessment, go to the course blackboard page, and click on the "Project 4" item, and then click on "Browse My Computer" and upload your final files (project4.py, report4.pdf, project4.ipynb, video file). Finally, click "Submit".