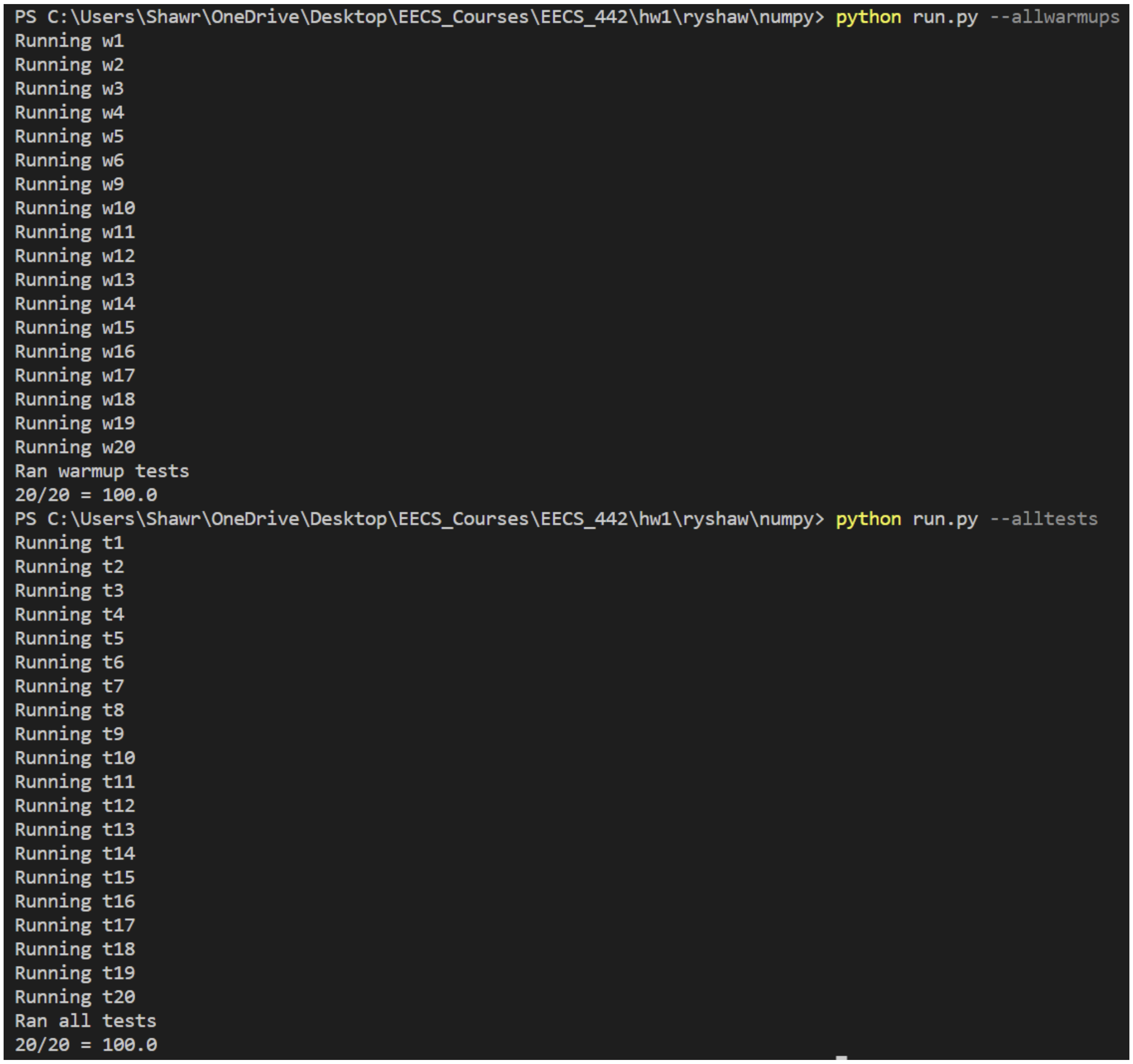
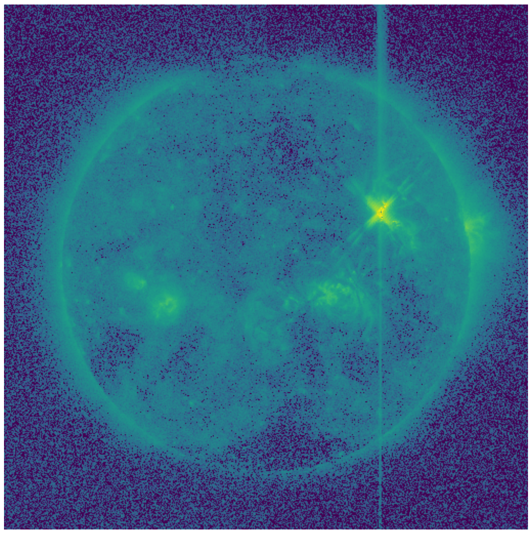
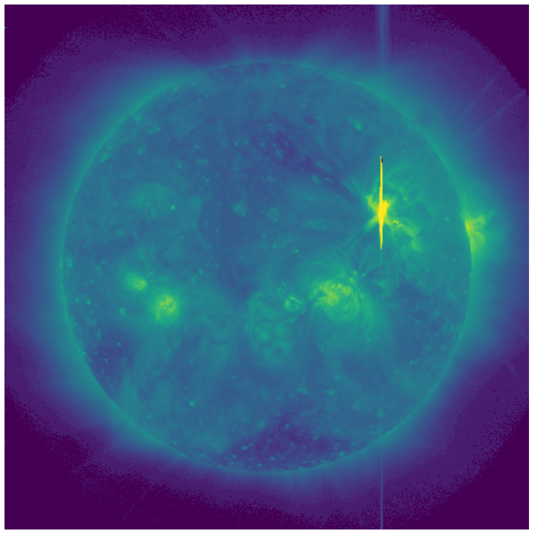
Ryan Shaw

Question 1.1 (the only question): Fill in the code stubs in tests.py and warmups.py. Put the terminal output in your pdf from:

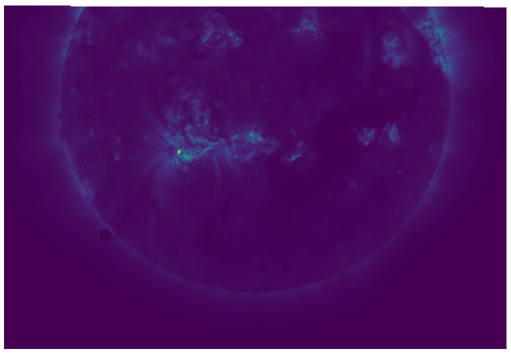
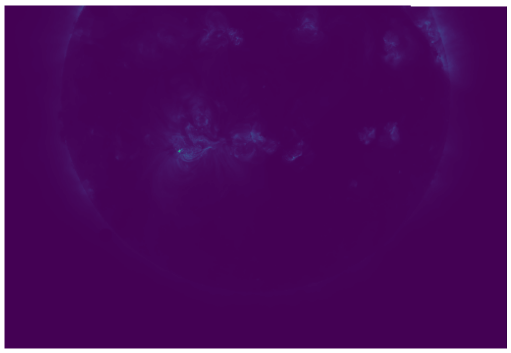


```
PS C:\Users\Shawr\OneDrive\Desktop\EECS_Courses\EECS_442\hw1\ryshaw\numpy> python run.py --allwarmups
Running w1
Running w2
Running w3
Running w4
Running w5
Running w6
Running w9
Running w10
Running w11
Running w12
Running w13
Running w14
Running w15
Running w16
Running w17
Running w18
Running w19
Running w20
Ran warmup tests
20/20 = 100.0
PS C:\Users\Shawr\OneDrive\Desktop\EECS_Courses\EECS_442\hw1\ryshaw\numpy> python run.py --alltests
Running t1
Running t2
Running t3
Running t4
Running t5
Running t6
Running t7
Running t8
Running t9
Running t10
Running t11
Running t12
Running t13
Running t14
Running t15
Running t16
Running t17
Running t18
Running t19
Running t20
Ran all tests
20/20 = 100.0
```

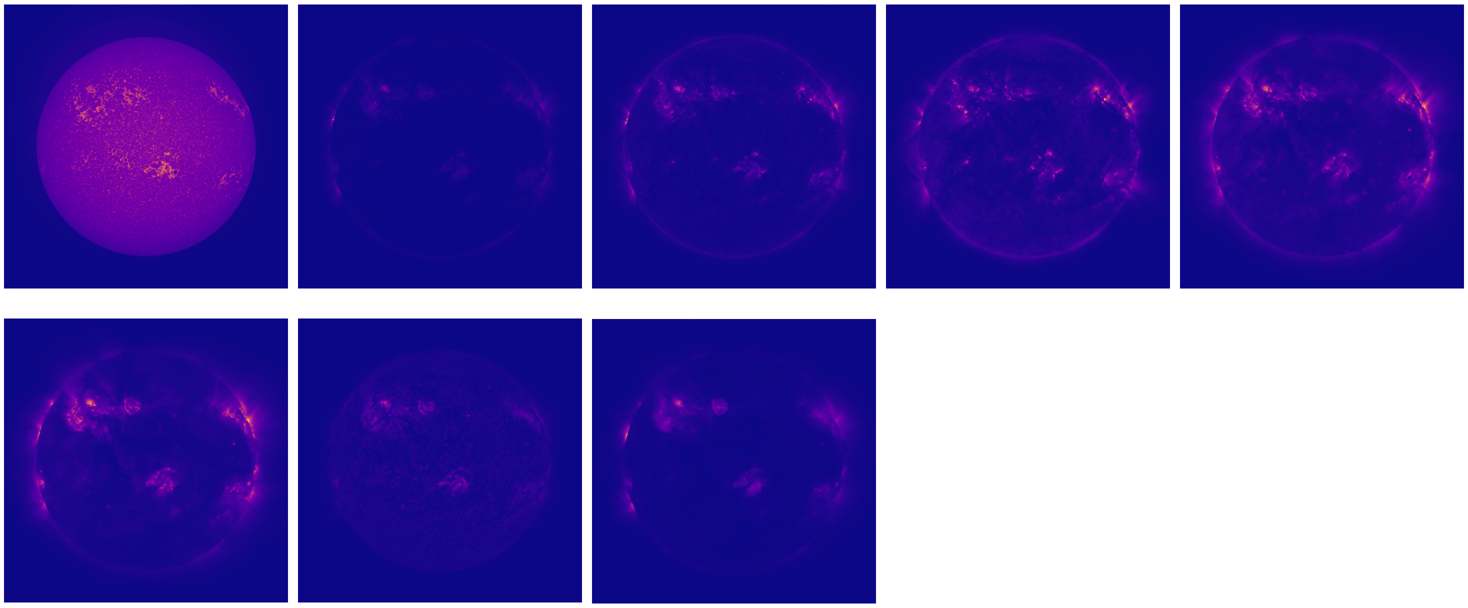
Report 3.1 (pictures, 2 points)

Report 3.2 (some pictures, 2 points)

Report 3.4 (3 points):

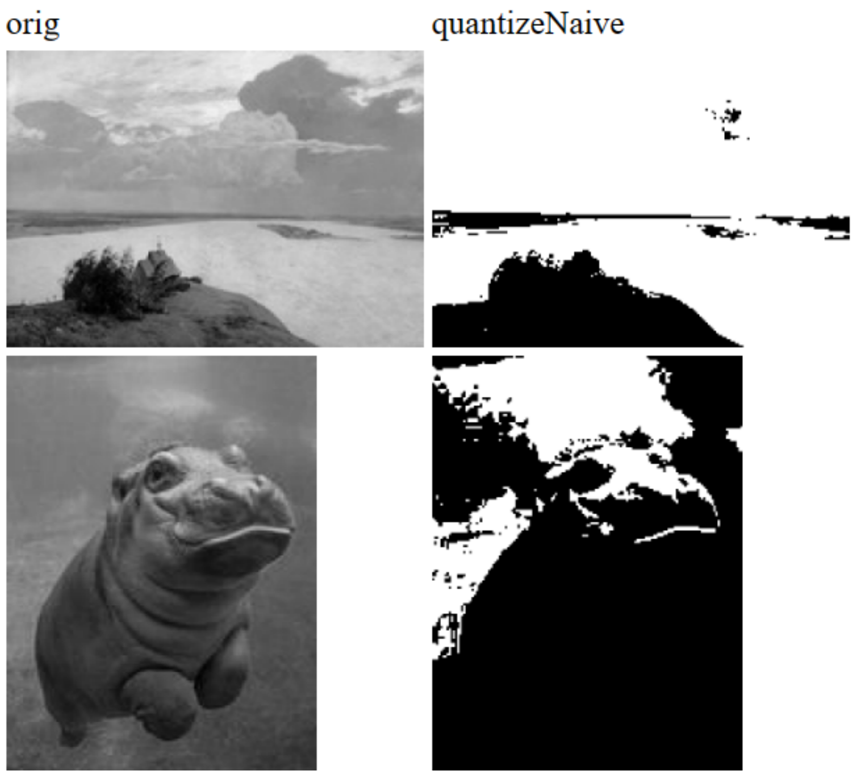


Report 4.1.3 (1 sentence, 2 points) If you apply this to the folder gallery, why might your code (that calls quantize) take a very long time?
This is because the algorithm is not vectorized, quantize() needs to be called on each pixel iteratively instead of concurrently. This results in a much longer computation time. Gallery also contains much sharper images (higher resolution). Meaning even more pixels would have to be processed than for gallery200

Report 4.1.4 (1 sentence, 2 points) Do low intensity values correspond to low palette values? Explain what's going on. You may have to look through the code you're given (a good habit to get into)

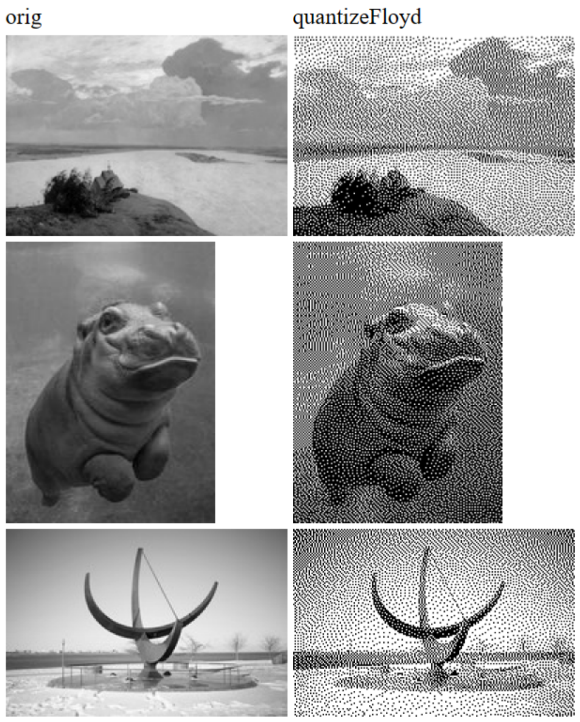
Since the quantized image finds the min difference between each pixel of the input image and the palette values. The palette has values ranging starting at 1 and ending at 0, meaning that higher indices result in lower palette values, or rather higher intensity values corresponding to lower indices. Meaning that ultimately the output will be somewhat inverted.

Report 4.1.5 (2 pictures, 4 points)

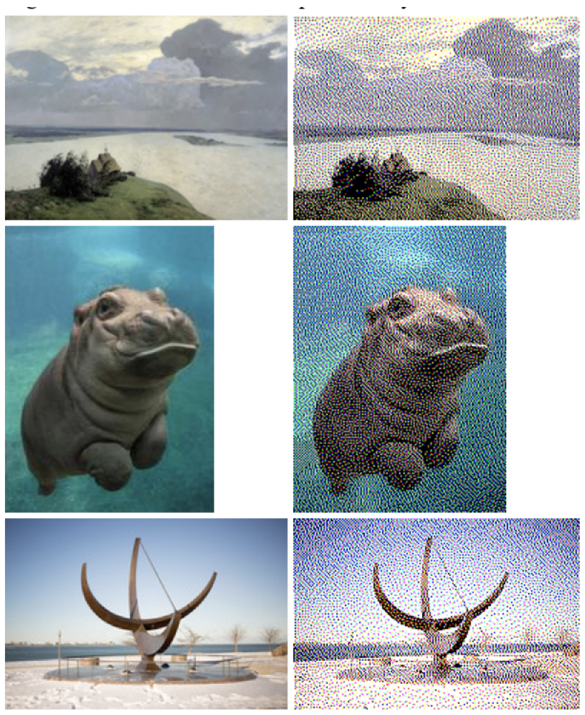


Report 4.2.2 (1-2 sentences, 3 points): In your own words, why does dithering (the general concept) work?
Try stepping back from your computer screen or, if you wear glasses, take them off.
Say I have a 8 bit color output, or lower resolution output, but the input data is much higher resolution, dithering allows me to seamlessly create that conversion by toggling numbits. I do wear glasses and I have a pretty bad prescription. When I take them off I see little difference in the resolution of the image.

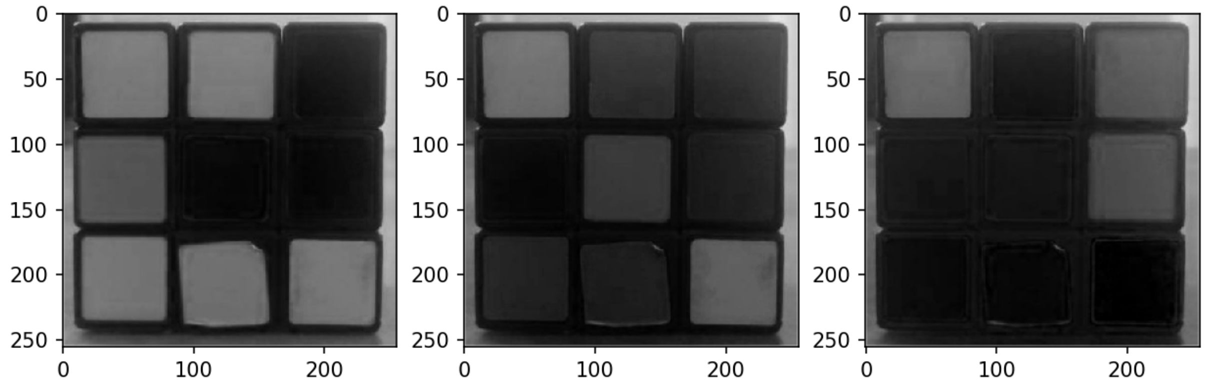
Report 4.2.3 (3 pictures, 12 points): Run the results on gallery200. Put three results in your document,
including aep.jpg. Don’t adjust --num-bits and use the defaults.



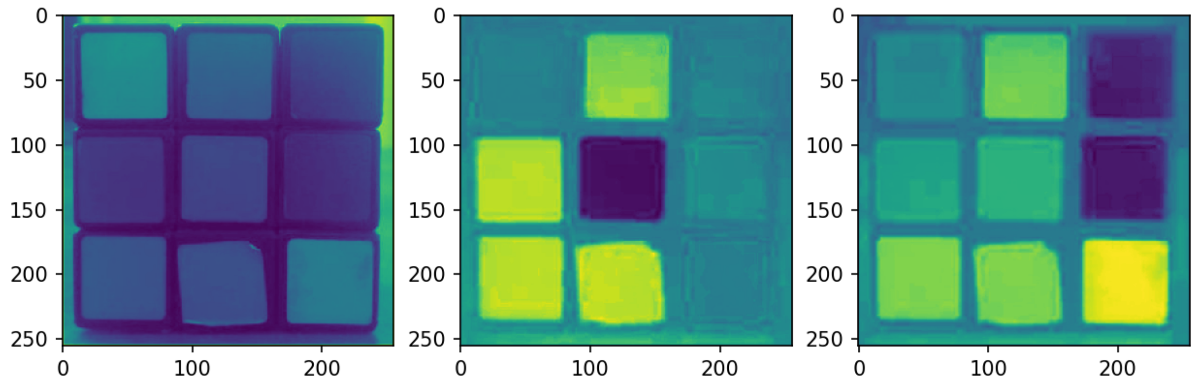
Report 4.4.3 (3 pictures, 5 points) Generate any three results of your choosing. This can be on the images we provide or on some other image you’d like. Put them in your document.



Coding 5.1: (three pictures, 2.5 pts) Load the images and plot their R, G, B channels separately as grayscale images using plt.imshow().



Coding 5.2: (three pictures, 2.5 pts) Then convert them into LAB color space using cv2.cvtColor and plot the three channels again.



Report 5.3: (5 pts) Include the LAB color space plots in your report. Which color space (RGB vs. LAB) better separates the illuminance (i.e., total amount of light) change from other factors such as hue? Why?

LAB is a better format to illustrate and separate illuminance from other factors such as hue, as it has a channel (L = lightness) dedicated solely to illuminance.