3460:415/515 Big Data Programming, Project 1 – Pooling

Project Description: Convolutional neural networks are often applied to analyze images. Pooling is a technique used in CNN for subsampling, reducing the dimensions of data. For this project, you'll use NumPy to practice pooling techniques. To facilitate the grading, you are ***required*** to use **NumPy**. NO other Python package except **numpy** and module **sys** (for obtaining command-line arguments) can be used. Submit your program in .py format. See the template for the project.

Input:

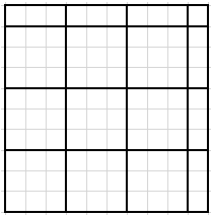
1) a gray scale image in .pgm (portable gray map) format. Your .pgm image file requires 4 entries followed by the greyscale values. The four entries are: the literal "P2", an integer representing the x dimension, an integer representing the y dimension, and an integer representing the maximum greyscale value. There should be x times y number of grey-level values after these 4 entries. Your .pgm files might contain lines of comments. The comment lines if any will start with the character # after the format literal "P2". Part of a sample plain pgm image (40 columns × 42 rows) bug.pgm is shown below.

```
P2
# a bug
# a small bug with a big hand
40 42
255
192 192 192 192 192 192 192 192 192 192 192 192 192 192 192 192 192
192 192 192 192 192 192 192 192 192 192 197 197 197 191 192 192 ...
```

The image of bug.pgm looks like: 

2) Pooling size: n (an integer, assuming a square window of size nxn)

Note: the .pgm file name and the pooling size along with Part number (see below) are to be passed to your program via command-line. Do not hard code the filename and pool size in your program.



Output: processed image.

Part 1. Maximum pooling. For this part, you implement the pooling technique as described in class. Note if your image size is not a multiple of pool size, your processed image should have the aspect ratio: ceil(image_width/pool size):ceil(image_height/pool size). See example pools shown on the left. Make sure your implementation is efficient, i.e. use NumPy not Python list. If your input name is x.pgm (know that the x is a variable, it could be "bug", "flower", …), save your processed file with correct header as: x_pooled_n.pgm where x is the image file name and n is the pool size. Make sure we can view the processed image using a pgm viewer.

Part 2. Oil painting your image. Using median pooling and moving your pooling window continuously across each pixel generates an image that looks "somewhat" like an oil painting. For this part, implement oil-painting using NumPy. If your input name is x.pgm, save your processed file with correct header as: x_oil_painted_n.pgm. Make sure we can view the processed image using a pgm viewer.

Part3 (Graduate students only). Oil painting a colored image. The image files will be in .ppm format. Assume no comment lines. The median pooling will be done in each of the three different color channels (RGB) separately. If your input name is x.ppm, save your processed file with correct header as: x_oil_painted.ppm.

Submission instructions. Submit an electronic copy of the program using the dropbox at Brightplace. I am expecting just one file. Name it: ***yourUAnetID_project1.py (ex: duan_project1.py)***

Be sure to submit your working solution before the due date! Do not submit non-working programs. The electronic submission time will be used to assess late penalties.