CS 6643 **E. K. Wong**
Project 2: Human Detection **Fall 2018**

Total # of points = 100.

Project Description. In this project, you will implement the *Histograms of Oriented Gradients (HOG)* feature and a *Two-Layer Perceptron* (neural network) for detecting human in 2D color images. You first need to convert the color image into a grayscale image using the formula $I = Round(0.299R + 0.587G + 0.114B)$ where *R, G* and *B* are the pixel values from the red, green and blue channels of the color image, respectively, and *Round* is the round off operator.

Use the Prewitt's operator to compute *x* and *y* image gradients from the grayscale image and then compute edge magnitude and gradient angles. Compute the edge magnitude by using the formula $M(i,j) = \sqrt{G_x^2 + G_y^2}$ where G_x and G_y are the horizontal and vertical gradients. Normalize and round off the results to integers within the range [0,255]. Compute the gradient angle by using the formula $\theta = \arctan(\frac{G_y}{G_x})$. The computed gradient angle should be with respect to the positive *x* axis that points to the right. For those locations where the templates go outside of the border of the image, you can let the gradients be undefined and assign a value of 0 to both edge magnitude and gradient angle. Assign a value of 0 to both edge magnitude and gradient angle if G_x and G_y are both 0.

Refer to lecture notes on how to compute the *HOG* feature. Use the unsigned representation and quantize the computed gradient angle into one of the 9 bins as shown in Table 1 below. If the gradient angle is in the range [170, 350) degrees, simply subtract by 180 first. Use the following parameter values in your implementation: *cell size* = 8 x 8 pixels, *block size* = 16 x 16 pixels (or 2 x 2 cells), *block overlap* or *step size* = 8 pixels (or 1 cell.) Use *L2* norm for block normalization. Leave the histogram and final descriptor values as floating point numbers. Do not round off to integers.

Design a two-layer perceptron for classifying the images represented by their HOG descriptor into *human* or *no-human.* The perceptron will have an input layer of size *N*, with *N* being the size of the HOG descriptor, a hidden layer and an output layer with one output neuron. Try hidden layer sizes of 250, 500 and 1,000 neurons and report the classification results. (Optional: you can try other hidden layer sizes and report the results if you get better results than the three above.) Use the *ReLU* activation function for neurons in the hidden layer and the *Sigmoid* function for the output neuron. The Sigmoid function will ensure that the output is within the range [0,1], which can be interpreted as the probability of having detected *human* in the image. Use the backpropagation rule we covered in lecture for training multi-layer perceptrons.

You can use Python, C++/C, Java or Matlab to implement your program. If you plan on using another language, send me an email first. You are not allowed to use any built-in library functions for any of the steps that you are required to implement, including the computation of HOG feature and the backpropagation training of neural networks. The only library functions you are allowed to use are those for the reading and writing of image files, matrix and vector arithmetic, and certain other commonly used mathematical functions.

Training and test images: A set of 20 training images, containing 10 positive (human) and 10 negative (no human) samples, and a set of 10 test images containing 5 positive and 5 negative samples will be provided. You should assign an output label of 1 for training images containing human and 0 otherwise.

Hand in: (a) Your source code. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have this. (b) Output results on *.txt* files (details will be provided later.) (c) An MS Words or PDF report that contains: (i) File names of your source code and output files. (ii) Instructions on how to compile and run your program. (iii) Output results (details will be provided later.) (iv) The source code of your program (Copy-and-paste over the source code onto the report.)

Table 1. Histogram Bins

Bin #	Angle in degrees	Center
1	[-10,10)	0
2	[10,30)	20
3	[30,50)	40
4	[50,70)	60
5	[70,90)	80
6	[90,110)	100
7	[110,130)	120
8	[130,150)	140
9	[150,170)	160