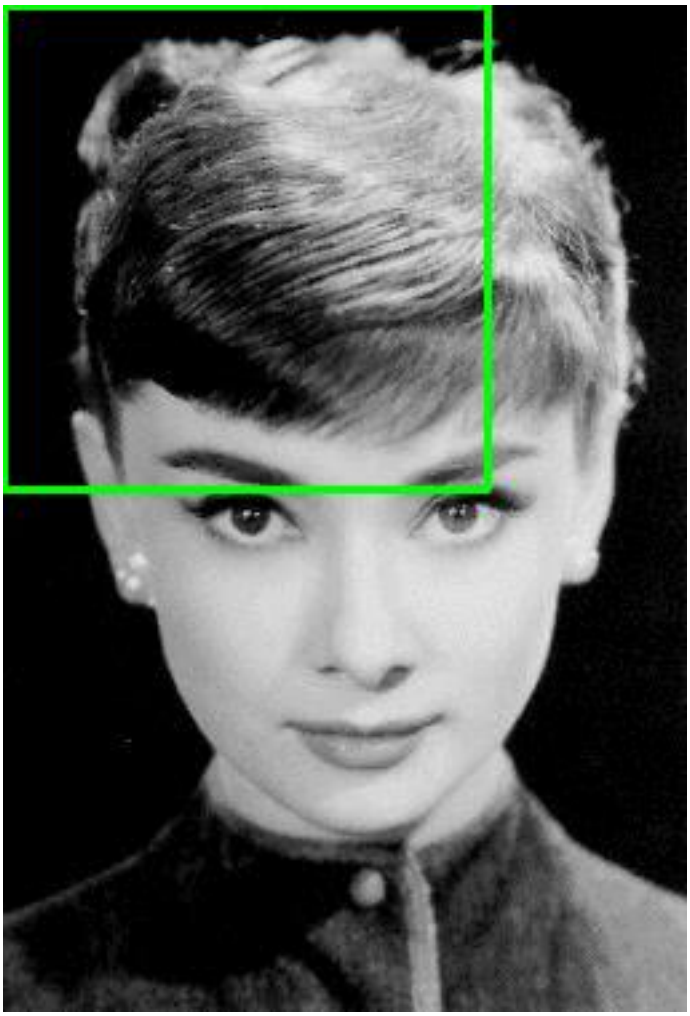
Face Detection with Histogram of Gradients

In this project, you are asked to build a head detection algorithm using the sliding window based histogram of gradients method proposed by Dalal and Triggs in 2005. In the project you are provided with a list of images containing faces and a list of images not containing faces. You will implement a hog descriptor and using SVM classifiers, learn to predict if a face is present in a given image.

Sliding Window Based Object Detection:

A sliding window is a rectangular region of fixed width and height that "slides" across an image. An example can be seen in the following figure.



For each window, a classifier can be run to determine whether the encapsulated region in the rectangle contains an object or not. In this project, we will be implementing a sliding window approach to detect faces.

The sliding window takes several parameters such as **width** and **height** of the rectangle and **stepsize**. Stepsize indicates how many pixels we are going to skip when iterating the sliding window in the x and y directions. While having a smaller stepsize increases detection accuracy, it also slows down our method. In practice, stepsizes like 4-8 are often determined to yield optimal performance in terms of balancing accuracy and speed.

The sliding window approach can be run at different scales (also called image pyramid method) to perform face detection at multiple scales. An “image pyramid” is a ***multi-scale representation*** of an image.

Utilizing an image pyramid allows us to **find objects in images at different scales** of an image. And when combined with a **sliding window** we can find objects in images in various locations.

At the bottom of the pyramid we have the original image at its original size (in terms of width and height). And at each subsequent layer, the image is resized (subsampled) and optionally smoothed (usually via Gaussian blurring).

What is HOG?

Histogram of oriented gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image or a detection window.

Given an image the HOG algorithm can be summarized as follows:

1. Divide the image into small connected regions called cells. Cellsize is given as a parameter to the hog calculation method.
2. For each cell compute a histogram of gradient directions or edge orientations for the pixels within the cell.
3. Create a histogram by discretizing each cell into angular bins according to the gradient orientation.
4. Each cell's pixel contributes according to its weighted gradient to its corresponding angular bin.
5. Groups of adjacent cells are considered as spatial regions called blocks. Blocksize is also a parameter for Hog. All the histograms belonging to cells in the same block are added together to form a feature representation for the given block.
6. The block histogram is normalized and is ready to be used as a feature descriptor representing the contents of the block.

Details of assignment and What to Impelement:

The provided face images are 36 x 36 pixel images of aligned faces. However the background images, like the test images are larger and heterogenous in size. To handle those images, you will be implementing a sliding window based processing pipeline. In the project you will be responsible for implementing the following methods:

1. main.py: The top level script for training and testing your sliding window based object detector.
2. hog.py: The method where the histogram of gradients calculation from Images is done. You will have two methods: extractHogFromImage and extractHogFromRandomCrop. You will use extractHogFromImage to get positive class features and extractFromRandomCrop to get negative class features.
3. classifier_train.py: Trains a linear SVM classifier using HOG features to seperate images containing faces from images that do not contain faces (Use sklearn.svm.LinearSVC or a similar method for classification).
4. object_detect.py Runs the classifier using a sliding window approach on the test set. Using a scale parameter, for each image, object_detect.py runs the object detection algorithm at multiple scales and performs non-maxima suppression to remove duplicate detections.

5. evaluation.py: Computes average precision and mean Intersection over Union score for each image and the entire validation dataset.
6. visualization.py: Visualizes detections on each image and displays / saves the entire validation set in a loop. On the same image, draw ground truth bounding boxes as red and predicted bounding boxes as green.

Data:

The training and validation datasets can be downloaded from the following link. https://www.dropbox.com/s/83q3vezvxu1inmm/Data.zip?dl=0 . In the downloaded folder, there are 3 sets of data, namely (FaceImages, NonFaceImages and a ValidationSet). Please do not submit these images back with your assignment submission.

Code:

The development for the code should be done in python 3. You are recommended to use the numpy, opencv and scikit libraries. For the image manipulation tasks such as hog calculation, you are not allowed to use pre-made functions for gradient, orientation and histogram calculations.

- When coding your HOG features, you are encouraged to play with hog and sliding window parameters. To give a starting point, HoG cell sizes of 6 are reasonable. Positive patch sizes of 36 x 36 pixels are reasonable. Negative patch sizes of 36 x 36, 48 x 48, 72 x 72, sampled with step sizes (sliding window stride) of 48 are reasonable as a starting point. On the evaluation set, a step size of 4 is a good starting point.
- If training takes long, **work on a smaller training set** while writing and debugging your code. You dont want to wait minutes each time you debug your code.
- Your code will be **checked for plagarism** with both your friends code and with popular online implementations of this method. If you take code snippets from an online source, give credit both in your code file and in your report to avoid a plagarism penalty.

Report:

- Describe the algorithm you implemented.
- Show and discuss the results of your algorithm. Present the best parameters for hog and sliding window pipeline. You should experiment with at least two different hog cell size, hog block size, sliding window size and evaluation stride. Compare each parameter individually, holding the other parameters constant and report the resulting average precision values.
- Show how your method performs on 3 images you took of yourself (In one of them look directly at the camera, in one of them put your hand in front of your mouth and in the last one look at somewhere other than the camera).
- Discuss any extra credit you did and clearly show its contribution on the results.

Grading:

- 30 pts Hog Calculation
- 15 pts Linear Classifier Training
- 20 pts Sliding Window Object Detector
- 15 pts Multiscale object detection and non-maxima suppression.

- 30 pts Report with and requested results and evaluation of multiple parameters .
- 20 pts Extra Credit.

Extra Credit:

- (Up to 10 pts) Experiment with alternative features and / or classifiers
- (Up to 5 pts) Find and use alternative positive and negative training data.
- (Up to 10 pts) Any other creative solutions to improve baseline results.

Submission:

Submit your report as a pdf file and your code as .py files with the names we provided. They should all be zipped in a file with the name **NAME_SURNAME_CVHW4.zip.**

For questions, email alpkindiroglu@gmail.com.

Credits:

This project was originally created by James Hays of Georgia Tech and also given as an assignment at University of Texas. The datasets used in the project is identical and was gathered for those assignments.