SCC0251 + MAI5020 — Prof. Moacir Ponti

PAEs: Gustavo Evangelista Araújo and Leo Sampaio Ferraz Ribeiro

Assignment 03: image descriptors

Code the assignment by yourself. Ask if you need help. Plagiarism is not tolerated.

1 Introduction

1.1 The Task

In this assignment you have to implement a classic feature extractor called Histogram of Oriented Gradients (HoG). It was first introduced by Dalal and Triggs [1] for human detection in CCTV images. We will follow in their steps and implement a solution for the same task using both the descriptor and the Machine Learning algorithm K-nearest-neighbours. Read the instructions for each step. Use `python 3` and the libraries `numpy`, `imageio` and `scipy` to complete the task.

Follow the instructions carefully:

1. **Read the parameters**
2. **Load** the three image collections $X_0 = \{p_0, \cdots, p_n\}$, $X_1 = \{p_0, \cdots, p_m\}$ and $X_{test} = \{t_0, \cdots, t_o\}$, containing images without humans (X_0), images with humans (X_1) and test images (X_{test}).
3. **Transform all images** to black&white using the Luminance technique
4. **Implement and compute the HoG descriptor** of all images.
5. **Implement and Train a K-nearest-neighbours** model with $K = 3$ using the training images X_0 and X_1.
6. **Predict** the classes for each of the test images X_{test}.
7. **Print** the predictions separated by a single space.

1.2 Input Parameters

The following parameters will be input to your program in the following order through `stdin` (standard input):

1. one line with list of space-separated filenames for training images without humans in them,
2. one line with list of space-separated filenames for training images with humans in them,
3. one line with list of space-separated filenames for testing images

2 Preprocessing

Before computing the image descriptors we need to transform the images to black&white. For this task we are are going to use the common *Luminance* method, for each RGB pixel:

$$f(x, y) = \lfloor 0.299R + 0.587G + 0.114B \rfloor,$$

where $\lfloor . \rfloor$ is the floor operation and letters R, G, B represent each colour channel.

3 Histogram of Oriented Gradients

This descriptor is a good way of capturing how the textures in an image are "arranged" by looking at the angle of the gradients (the *rate-of-change* from one pixel to the next at a certain angle). To compute this we must first obtain the gradient of the image in both x and y directions. This can be done by performing convolution (use `scipy.ndimage.convolve`) of an image with the sobel operator in each direction:

$$w_{Sx} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} ; \; w_{Sy} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

After computing the convolution between the image and each filter, we will have the image gradients g_x and g_y. Using those matrices, we can separate the gradient's magnitude and angle into two other matrices. The magnitude can be computed:

$$M(x, y) = \frac{\sqrt{g_x(x, y)^2 + g_y(x, y)^2}}{\sum_x \sum_y \sqrt{g_x(x, y)^2 + g_y(x, y)^2}}$$

and the angles:

$$\phi(x, y) = \tan^{-1} \frac{g_y(x, y)}{g_x(x, y)}$$

Now, we need to use those angles to accumulate magnitudes in a binned fashion. First, **(i)** sum $\pi/2$ to all values in ϕ to make the angle range shift from $[-\pi/2, \pi/2]$ to $[0, \pi]$, then, **(ii)** convert the angles from radians to degrees (see `np.degree`); **(iii)** digitise the angles into 9 bins, this means slicing the angle range in 20 degree intervals $[0, 19], [20, 39], ...$ and then for each $\phi(x, y)$, determine in which of the 9 bins the value falls into.

We want to use the bins determined for each x, y position to accumulate the magnitudes. Let's consider an example with post-shift and conversion matrix:

$$\phi = \begin{bmatrix} 10 & 23 & 34 \\ 43 & 12 & 91 \\ 0 & 92 & 1 \end{bmatrix}$$

And the magnitude matrix:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

Computing the bins for each position we would have:

$$\phi_d = \begin{bmatrix} 0 & 1 & 1 \\ 2 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix}$$

where, for example, $\phi_d(1,2) = 4$ because $\phi(1,2) = 91$ falls into the range $[80, 99]$, which is bin 4. Now, using this setup to accumulate magnitudes using the bins, we'd have a resulting descriptor:

$$d_g = \begin{bmatrix} 7 & 5 & 3 & 0 & 3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where, for example, $d_g(1) = 5$ because positions $\phi_d(0,1)$ and $\phi_d(0,2)$ are in bin 1 and the magnitude for those positions is $M(0,1) = 2$, $M(0,2) = 3$.

This descriptor is then a vector with 9 dimensions, d_g.

NOTE: In this example there will be some divisions by zero that will result in infinity. This is ok as `np.arctan` works with infinity. Your code will however produce some warnings that are flagged by the auto-grader. You can silence the warnings with `np.seterr(divide='ignore', invalid='ignore')`.

4 K-Nearest Neighbours

Just as the Histogram of Oriented Gradients is a classic of image feature extractors, K-Nearest Neighbours (KNN) is a classic of Machine Learning literature. The concept for KNN is very simple: there is no training step to the model and inference/prediction is done by comparing new samples against the training set

With a selection of labelled feature vectors (which we have in the form of the features extracted from X_0 and X_1), prediction with KNN consists of comparing a new sample (a feature vector from an image in X_{test} for instance) against all of the vectors in the labelled training set. For doing such comparison we will make use of euclidean distance between the vectors:

$$D(a,b) = \sqrt{\sum_x (d_g(a) - d_g(b))^2}$$

where a and b are images and d_g is the HoG feature extractor.

After computing the euclidean distance between a test sample and all of the training sample, the predicted class (has human or does not) is determined by a "voting system". The K (here defined as $K = 3$) training samples closest to the test sample each get "a vote", which is determined by their original class. For example, if 2 of the closest samples came from X_0 and one came from X_1, then the majority of votes goes to class 0, meaning that no human was detected in the test sample (the predicted label is 0).

Your goal then is to run this prediction algorithm for each of the test samples in X_{test} and print out the resulting predictions for weather there is a human (1) or not (0) in each image. The output of your code consists of printing out the predictions for each test sample separated by a single space.

5 Input and Output

Input Example 01. 10 images without humans, 10 images with humans and 10 images for test are input to your code:

```
0/0.png 0/1.png 0/2.png 0/3.png 0/4.png 0/5.png 0/6.png 0/7.png 0/8.png 0/9.png
1/0.png 1/1.png 1/2.png 1/3.png 1/4.png 1/5.png 1/6.png 1/7.png 1/8.png 1/9.png
0/0.png 0/1.png 0/2.png 0/3.png 0/4.png 0/5.png 0/6.png 0/7.png 0/8.png 0/9.png
```

Output Example 01. The result of using the HoG descriptor of the 20 input training images to train a KNN model and then using this model to predict whether the 10 testing samples have humans or not:

```
0 0 1 1 0 0 0 0 0 0
```

6 Submission

Submit your source code to e-disciplinas (only the `.py` file). You can check for correctness by downloading the test cases from e-disciplinas and testing with run-codes-local, which will be used by the PAEs to grade your work.

1. **Use your USP number as the filename for your code.**
2. **Include a header**. Use a header with name, USP number, course code, year/semester and the title of the assignment. A penalty on the evaluation will be applied if your code is missing the header.
3. **Comment your code**. For any computation that is not obvious from function names and variables, add a comment explaining.
4. **Organize your code in programming functions**. Use one function for the HoG feature extractor and one for the KNN algorithm.

7 Grading

Your work will be graded as:

$$0.3R + 0.5A_1 + 0.2A_2 - P$$

where each value ranges from $0-10$, R is the grade from run-codes-local, A_1 and A_2 refer to the correct implementation of the HoG descriptor and KNN respectively. P goes up to 1.0 and is a possible penalty for failing to follow the rules from the previous section.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.