Assignment 3:
Face Detection With A Sliding Window
COMP 408

Ahmet Uysal

November 28, 2018

Instructor: Yücel Yemez

1 Getting Training (HoG) Features

To train our Support Vector Machine classifier we need to get HoG samples from many positive and negative face samples. HoG features are calculated with the help of VL Feat toolbox. Positive face images from two different databases are used to get HoG features for face images. HoG features for negative samples are calculated from non-face images of random parts from given non-face image set, in different scales.

1.1 Positive Face Samples

1.1.1 Caltech Face Samples

Positive training database of 6,713 cropped 36x36 faces from Caltech Web Faces Project was given with the started code.

1.1.2 LFW Face Samples

13,233 face images from Labeled Faces in the Wild database are also used as positive samples. Original provided pictures was 250x250 and colorful. These images are converted to gray-scale and 36x36 with a MATLAB script.

```
data_path = '../data/';
lfw_faces_path = fullfile(data_path, 'lfw_faces');
image_files = dir( fullfile( lfw_faces_path, '*.jpg') );
num_images = length(image_files);

for i = 1:num_images
   img = imread(strcat(image_files(i).folder, '\',...
        image_files(i).name));
    if size(img, 3) == 3
         img = rgb2gray(img);
   end

   img = imresize(img, [36, 36]);
   imwrite(img, (fullfile(lfw_faces_path, image_files(i).name)))
end
```

Code Snippet 1: MATLAB script for processing images taken from LFW database.

1.1.3 Samples Produced by Warping

1.1.4 MATLAB Implementation

```matlab
image_files = dir( fullfile( train_path_pos, '*.jpg'));
num_images = length(image_files);

first_img = imread(strcat(image_files(1).folder, '\', image_files(1).name));
face_images = zeros([size(first_img), num_images], 'like', first_img);

% store the images to use in warping
for i = 1:num_images
    face_images(:, :, i) = imread(strcat(image_files(i).folder, '\',...
        image_files(i).name));
end
```

Code Snippet 2: MATLAB script for processing images taken from LFW database.

1.2 Negative Face Samples

These samples are taken from provided non-face images. Images are randomly chosen from all non-face images and samples are taken from five different scales (0.25, 0.5, 1, 1.5 and 2). Random hog_template_size × hog_template_size parts of the scaled images are taken and used in the HoG calculation. Images are turned to gray-scale since all of our positive samples are in gray-scale. In each scaled versions of the randomly selected image $\lfloor\sqrt{w * h/hog\_template\_size^2}\rfloor$ windows are taken from the image where w and h represents dimensions of the image. I added this part to take samples depending on the size of the images since the differ in size.

1.2.1 MATLAB Implementation

```matlab
image_files = dir( fullfile( train_path_neg, '*.jpg' ));
num_images = length(image_files);
num_samples = 65000;
% counter for stopping at num_samples
sample_count = 0;
% different scales used for extracting, can be changed
scales = [.25, .5, 1, 1.5, 2];
% initialize negative features matrix with zeros
features_neg = zeros(num_samples, (hog_template_size / hog_cell_size)^2 * 31);
% get num_samples samples
while sample_count < num_samples
    % randomly select and image to sample windows from given dataset
    rand_img_index = random('unid', num_images);
    rand_img = imread(strcat(image_files(rand_img_index).folder,...
        '\', image_files(rand_img_index).name));
    % convert selected image to grayscale if it's rgb
    if size(rand_img, 3) == 3
        rand_img = rgb2gray(rand_img);
    end
    % take windows for different scales of randomly selected image
    for scale = scales
        % scale the image
        rand_scaled_img = imresize(rand_img, scale);
        % take the dimensions of the image
        [w, h] = size(rand_scaled_img);
        % if dimensions are smaller than cell size we can't get any sample
        if w < hog_template_size || h < hog_template_size
            continue
        end
        % determine how many samples will be taken
        num_sample_from_img = floor(sqrt(w * h / hog_template_size^2));
        % take num_sample_from_img  samples from the scaled image
        for i = 1:num_sample_from_img
            % randomly select window location
            window_x = random('unid', w + 1 - hog_template_size);
            window_y = random('unid', h + 1 - hog_template_size);
            % increase the sample count
            sample_count = sample_count + 1;
            % calculate HoG for selected window
            hog = vl_hog(im2single(rand_scaled_img(...
                window_x:window_x+hog_template_size-1,...
                window_y:window_y+hog_template_size-1)), hog_cell_size);
            % add result to features_neg
            features_neg(i, :) = hog(:);
            % Stop if we react the wanted amount
            if sample_count >= num_samples
                break
            end
        end
        % Stop if we react the wanted amount
        if sample_count >= num_samples
            break
        end
    end
```

Code Snippet 3: Related part of the get_training_features function for getting HoG values of non-face images.

2 Training Support Vector Machine using Features

VL Feat toolbox is also used here to train a support vector machine, using positive and negative HoG features.

2.1 MATLAB Implementation

```
function svmClassifier = svm_training(features_pos, features_neg)
% INPUT:
% . features_pos: a N1 by D matrix where N1 is the number of faces and D
%   is the hog feature dimensionality
% . features_neg: a N2 by D matrix where N2 is the number of non-faces and D
%   is the hog feature dimensionality
% OUTPUT:
% svmClassifier: A struct with two fields, 'weights' and 'bias' that are
%       the parameters of a linear classifier

% combine the features in one matrix to give to vl_svmtrain
X = [ features_pos ; features_neg ];
% create the label vector for indicating positive or negative feature
Y = [ones(1, length(features_pos)) , -ones(1, length(features_neg))];
lambda = 0.00001;
% Function wants an D by N matrix (D: feature dimensions, N: feature count)
% so input the transpose of X
[w, b] = vl_svmtrain(X', Y, lambda);
svmClassifier = struct('weights',w,'bias',b);
end
```

Code Snippet 4: My implementation of svm_training function.

3 Testing SVM Classifier on Training Data

4 Analyze Classifier Performance on the Test Data using Sliding Windows

5 Dimensionality Reduction using PCA