Computer Vision Assignment Report

Title: Face detection with a sliding window

Student Name: 加泽瑄

Student ID: 12011126

1. Experimental Design

We totally have six sections for this assignment. And the report will follow that steps and show the corresponding result.

1.1 Load positive training crops and random negative examples

```
def get_positive_features(train_path_pos, feature_params):
    win_size = feature_params.get('template_size', 36)
    cell_size = feature_params.get('hog_cell_size', 6)

    positive_files = glob(osp.join(train_path_pos, '*.jpg'))

    n_cell = np.ceil(win_size / cell_size).astype('int')
    feats = np.random.rand(len(positive_files), n_cell * n_cell * 31)
    i = 0
    for path in enumerate(positive_files):
        temp_img = load_image_gray(path)
        img_feat = vlfeat.hog.hog(temp_img, cell_size)
        feats[i, :] = img_feat.flatten()
        i += 1

    return feats
```

```
def get_random_negative_features(non_face_scn_path, feature_params,
num_samples):
    win_size = feature_params.get('template_size', 36)
    cell_size = feature_params.get('hog_cell_size', 6)

    negative_files = glob(osp.join(non_face_scn_path, '*.jpg'))

    n_cell = np.ceil(win_size / cell_size).astype('int')
    feats = np.random.rand(len(negative_files), n_cell * n_cell * 31)
    all_feat = []

    for path in negative_files:
        temp_img = load_image_gray(path)
        # change size into win_size
        w, h = temp_img.shape
```

```
        for y in range(0, int(w / int(win_size) - 1)):
            for x in range(0, int(h / int(win_size) - 1)):
                img = temp_img[y * win_size:(y + 1) * win_size, x * win_size: (x
+ 1) * win_size]
                img_feat = vlfeat.hog.hog(img, cell_size)
                all_feat.append(img_feat.flatten())

    # randomly get negative features
    rand_numbers = random.sample(range(0, len(all_feat)), len(negative_files))
    for i in range(len(negative_files)):
        feats[i, :] = all_feat[rand_numbers[i]]

    return feats
```

2.2 Train Classifier

```
def train_classifier(features_pos, features_neg, C):
    label = np.ones((len(features_pos) + len(features_neg)))
    label[len(features_pos):len(label)] = 0
    features = np.vstack((features_pos, features_neg))

    svc = LinearSVC(random_state=0, tol=1, C=C)
    svm = svc.fit(features, label)

    # svm = PseudoSVM(10, features_pos.shape[1])
    return svm
```

1.3 Mine Hard Negatives

```
def mine_hard_negs(non_face_scn_path, svm, feature_params):
    win_size = feature_params.get('template_size', 36)
    cell_size = feature_params.get('hog_cell_size', 6)

    negative_files = glob(osp.join(non_face_scn_path, '*.jpg'))

    n_cell = np.ceil(win_size / cell_size).astype('int')
    feats = np.random.rand(len(negative_files), n_cell * n_cell * 31)

    all_feat = []
    for path in negative_files:
        temp_img = load_image_gray(path)
        # change size into win_size
        w, h = temp_img.shape

        for y in range(0, int(w / int(win_size) - 1)):
            for x in range(0, int(h / int(win_size) - 1)):
                img = temp_img[y * win_size:(y + 1) * win_size, x * win_size: (x
+ 1) * win_size]
                img_feat = vlfeat.hog.hog(img, cell_size)
                all_feat.append(img_feat.flatten())
    feats = []
    for i in range(len(all_feat)):
        y_pred = svm.predict([all_feat[i]])
```

```
        if y_pred == 1.0:
            feats.append(all_feat[i])

    return np.array(feats)
```

2. Experimental Results

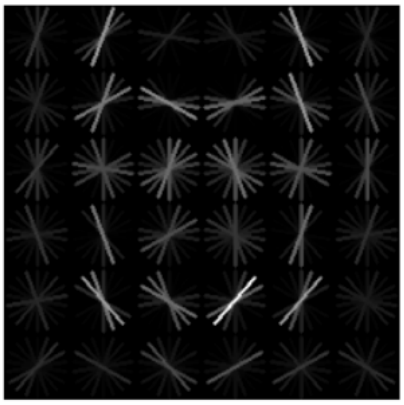
2.1 Train Results

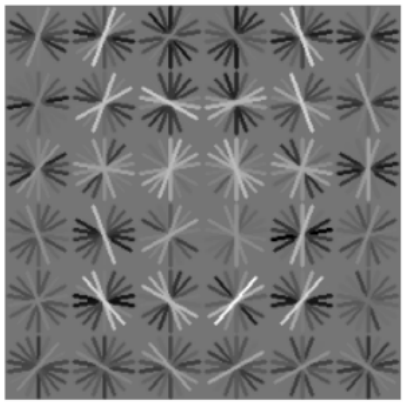
Positive training crops and random negative, the result of the first standard training.

```
Accuracy = 99.971%
True Positive rate = 99.985%
False Positive rate = 0.365%
True Negative rate = 99.635%
False Negative rate = 0.015%
```

Mine Hard Negatives, Improved training results

```
Accuracy = 99.728%
True Positive rate = 99.717%
False Positive rate = 0.000%
True Negative rate = 100.000%
False Negative rate = 0.283%
```





2.2 Evaluate and Visualize Detections