COMP9517 Individual Component

Victor Tse (z5075018)

I. Introduction

We are provided with a dataset in excess of 81,000 real-world colour images, where each image is of a standardised width and height (wide aspect ratio of 64x80 pixels respectively). Images have been categorically sorted into 'positive' and 'negative' binary labels, denoting whether the image contains pedestrians or not (respectively).

Given this dataset, we are tasked with designing a classifier in Python (OpenCV/scikit-learn packages) that will accurately label a similar image (of the same dimensions) as 'positive' or 'negative'. This will form the basic framework for a pedestrian detector in the group project. Deep learning techniques are not permitted for this task. As part of the experiment, a selective parameter sweep will also be done to better understand how the chosen detector method performs under different parameters.

II. Background & Method

A. Provided Dataset

Our pedestrian dataset [1] has been provided by National ICT Australia (NICTA). Key characteristics of this dataset include:

- The underlying source images used to generate any pair of provided training subsets are guaranteed to be alike. This suggests that some portion of the training subset should be enough for approximately representing most of the training set, if computational constraints need to be applied.
- The validation set has not 'polluted' with any of the training subsets, in that the validation images were not generated from the same source images that generated the training images. This suggests that validation can be done fairly for any subset of training that is used.
- Given the real-life street conditions of this dataset, individual images are expected to be highly variable, with one or more pedestrians in a variety of poses, azimuth orientations, colour/lighting levels, and with/without partial occlusions. Pedestrians are however expected to be at about the same relative scale (relative to the height of the image). Some images are also expected to be mirrored versions of other existing images.

B. Histogram of Oriented Gradients

For this task, I propose using the Histogram of Orientated Gradients (HOG) -based method for pedestrian detection. [2] Conceptually, HOG is simple to interpret, and in the interests of development time, well-supported with well-known examples online under the existing Python libraries. In addition, HOG is said to now be a "de facto standard" for many practical applications of computer vision due to its good performance [3]. As trade-offs however, HOG requires pedestrians to be in "more or less upright postures" with only partial occlusion at best.

The HOG method of detection works as follows [4]:

1. Gamma/colour normalisation is first applied by performing square root gamma compression on each colour channel to help reduce illumination effects.
2. The grad operator is applied as a mask/filter in the x and y directions of the image to compute its signed/unsigned gradients.
3. Spatially local regions of pixels are then grouped together to form equally sized *cells*. From this, a gradient histogram is created for each cell, with each bin representing a uniformly spaced angular orientation. Each pixel's gradient is placed into its corresponding bin.
4. Similarly, spatially local regions of cells are then grouped together to form equally sized *blocks*. Blocks may partially overlap with one another for better performance. Localised contrast normalisation is applied on each block to improve invariance to illumination, shadows and edge contrast.
5. The HOG descriptors are collected from each block to form a feature vector for that image. By applying Steps 1-5, a training set of images can thus be pre-processed into a collection of HOG feature vectors, with each one corresponding to its label (positive/negative).
6. A pedestrian detector classifier can then be constructed from this training set. In the original paper, a support-vector machine (SVM), using either a linear or Gaussian kernel, is applied for simplicity and speed. This constructs an SVM-based model that determines a hyperplane representing the most optimal decision boundary between the two positive/negative classes.
7. For each test image intending to be classified, Steps 1-5 must be repeated to first. convert the image to its HOG feature vector representation. This can then be inputted into the trained SVM model for classification.

Note that the training and test images are required to be identical in their pixel dimensions to ensure consistency in the feature vectors being produced. This constraint is assured by the NICTA dataset.

C. Other Classifiers

HOG is regarded as an old yet robust technique. In more recent years, boosting techniques such as Adaboost [5] can be used to combine/average the results of different classifiers for better overall speed and performance. References [6] and [7] provide an insight into the performance of up to 40 classifiers including deep-learning and state-of-the-art techniques that have been developed up until 2014.

III. Experimental Setup

A. Model Evaluation Environment

A Python script `z5075018_hogsvm_testbed.ipynb` has been developed which implements the HOG+SVM method of pedestrian detection described previously, under the scikit-learn library. The environment has been set up such that adjustable parameters under this method can be automatically scanned for under multiple user-specified values. The generated pre-processed images and model are also saved as pickle-serialised files in case of later post-analysis.

In aiming to better understand how parameters of the HOG method may affect the performance of the classifier, I restrict my analysis to looking at HOG parameters only, to reduce the dimensionality of the parameter sweep. Under these conditions, `LinearSVC()` is used for computational speed under large datasets, with fixed parameters matching the original paper: `‘hinge’` loss (standard SVM loss), and default values otherwise. Fixing the parameters of the linear SVM model may be reasonable under the assumption that “most progress in pedestrian detection can be attributed to the improvement in features alone” [7], or that it is the HOG feature vector that is the dominant factor in influencing the performance of the classifier.

Of the available adjustable HOG parameters provided by scikit-learn, I have chosen to restrict these further for computational reasons by always enabling square root gamma normalisation and fixing the block normalisation method to be its default (matching the original paper): `‘L2-Hys’`. Under these conditions, the significance of cell size (pixels/cell), block size (blocks/cell) and number of orientations will be explored. To further simplify, only square cells/blocks will be investigated. These decisions draw inspiration from “Figure 5” of [2].

For computational reasons, the size of the training set has also been fixed to utilise only 3000 positive and negative images each as an arbitrary value. Despite the relatively small training set size, this value was chosen as it was empirically found in early testing to already well exceed an accuracy of 90% on the provided test dataset.

B. Data Visualisation Environment

```
z5075018_hogsvm_textanalysis.py
z5075018_hogsvm_postanalysis.ipynb
```

To assist in the data analysis of the Results, a clunky but semi-automated environment has been developed through the additional above combination of scripts to combine the user-loaded CSV and text debug logs into generating several graphical plots, with assistance from pandas and matplotlib Python libraries.

C. Test Metrics

Standard statistical metrics have been provided as a general measure for the performance of this binary classifier. These include the accuracy, precision, recall/sensitivity, and area under curve (AUC) of the probabilistic receiver operating characteristic (ROC) for the classifier.

In the case of pedestrian detection, the ‘miss rate’ is commonly used as an evaluation metric, due to the consequences of failing to detect a pedestrian being extremely high (e.g. loss of life, by an automated vehicle). The miss rate is a measure for the number of false negatives that have occurred out of all occurrences that were explicitly positive, as described in Equation 1. Thus, this experiment will primarily focus on attempting to minimise the miss rate.

$$Miss\ Rate = \frac{FN}{TN+FP} \qquad (1)$$

D. Modifications to Original HOG Algorithm

It should be noted that in order to compute a probabilistic estimate for the ROC curve in scikit-learn for `LinearSVC()`, this object must be wrapped by a `CalibratedClassifierCV()` prior to training to perform cross-validation (at a default of five folds). This introduces a different validation approach to the HOG proposed in the original paper, where validation is done by recomputing the model with additional false positive training images explicitly being added to the training set. I expect the overall impact on the experiment to be limited.

As a side effect from applying cross-correlation, this may improve the overall model by reducing its likelihood to overfit.

IV. Results

The experiment was run, naively sweeping over all combinations of the parametric values in Table 1. This resulted in the construction of 352 separate models and feature vector generation iterations spanning multiple hours. Optimal parameters for miss rate, accuracy and AUC are also documented in Table 2.

TABLE I. Parameter Sweep Values

Parameter	Range
Block size (square)	1 – 4
Cell size (square)	4 – 14
Orientations	6 – 16

TABLE II. Parameter Optimisation Results

Optimised Parameter	Block Size	Cell Size	Orien.	Miss Rate (%)	Acc. (%)	AUC (%)
Miss Rate	4	4	14	0.8389	98.97	99.9438
Accuracy	3	4	14	0.9257	99.05	99.9466
AUC	3	4	16	1.012	98.90	99.9467

Assuming that the miss rate is to be optimised for, graphs have been generated under the optimal ‘miss rate’ parameters for trend analysis in Fig. 1-3. Fig. 4-5 showcase the most commonly occurring images in error across the 352 tests.

V. Discussion

Fig. 1 shows a general trend that as the cell size (square) is decreased from 14px to 4px, the miss rate, accuracy and AUC are all improved. This would suggest that, for this use case, HOG requires a lower cell size (for the resolution of the provided dataset) in order to accurately capture a detailed shape of a person’s features, such as their head and limbs. This does not

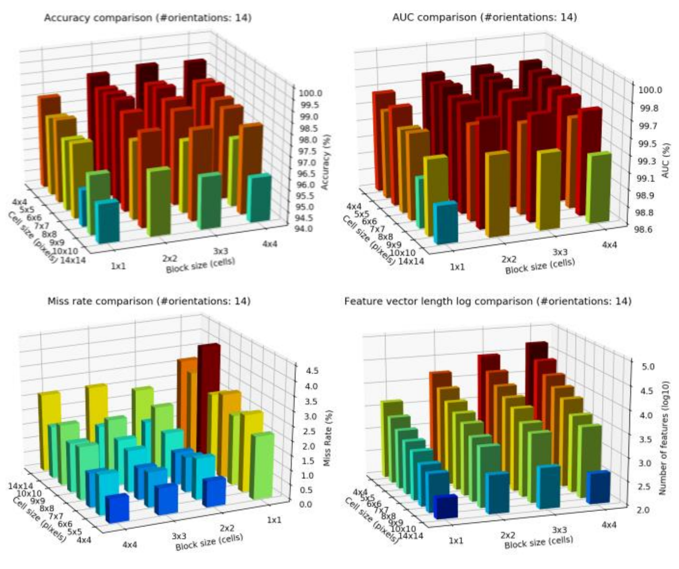

Fig. 1. Comparison of accuracy, AUC, miss rate, and feature vector length performance under varying cell and block sizes (fixed orientation size).

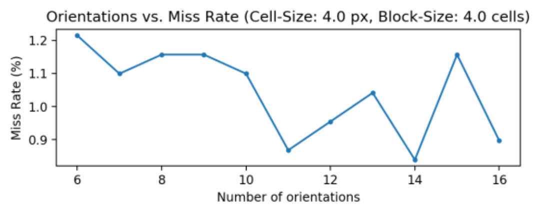

Fig. 2. Comparison of miss rate performance under varying orientations (fixed block and cell size).

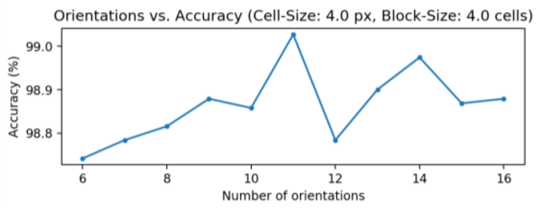

Fig. 3. Comparison of accuracy performance under varying orientations.

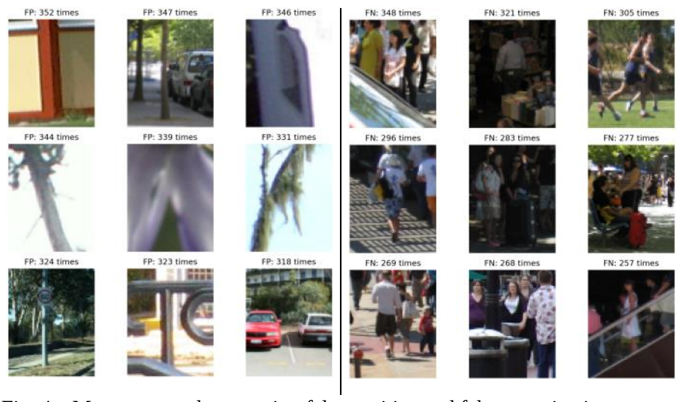

Fig. 4. Most commonly occurring false positive and false negative images.

appear to fully agree with that of the original paper, which suggested a cell size of 6-8 pixels as being optimal, with respect to their (different) dataset.

Fig. 1 also shows a general trend that as the block size (square) is increased from 1 cell to 4 cells, the miss rate, accuracy and AUC are all improved. This is expected, given that block normalisation is supposed to improve invariance to local details, perhaps by improving the ‘microscopic’ information of each cell using ‘macroscopic’ information. However, more analysis needs to be done at large block size values to validate the original paper’s intuitive claims that the results will begin to deteriorate again once the block size is made too big.

Fig. 1 also showcases (with a logarithmic vertical axis) that the feature vector will generally grow exponentially with either increasing block size or decreasing cell size. This draws a natural conclusion that the performance of our classifier will decrease if we attempt to improve the computational speed through only adjusting the parameters.

Although Fig. 1 has fixed the most optimal value of orientations to be 14, it can be (partially) seen in Fig. 2 and Fig. 3 that this result is not especially stable. A more comprehensive analysis and parameter sweep may still need to be required if clearer conclusions are to be drawn. Given that the chosen training set samples and sample size has not changed throughout any of the 352 tests conducted, I do not expect the trends of these graphs to change with increasing the training size.

Fig. 4 showcases the most commonly occurring false positives and false negatives out of the all tests conducted. Many false positive images can be seen to have strong vertical lines which are likely to cause confusion for the HOG method, given that upright pedestrian poses are a requirement for detection. Many false negatives may be attributed to occlusions to more than one-third of the pedestrian’s body, or possible confusion when a crowd of people are shown, all of whom are upright (creating noise).

References

[1] G. Overett, L. Petersson, N. Brewer, L. Andersson and N. Pettersson, “A new pedestrian dataset for supervised learning”, 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, 2008, pp. 373-378.

[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005).

[3] H. Bristow and S. Lucey, “Why do linear SVMs trained on HOG features perform so well?”, arXiv, 2014.

[4] scikit-image, “Histogram of oriented gradients”. https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html (Last accessed 1 Apr 2020)

[5] Y. Wei, Q. Tian and T. Guo, “An improved pedestrian detection algorithm integrating Haar-like features and HOG descriptors”, Advances in Mechanical Engineering, 2013. https://doi.org/10.1155/2013/546206

[6] P. Dollar, C. Wojek, B. Schiele and P. Perona, “Pedestrian detection: an evaluation of the state of the art”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, No. 4, April 2012, pp. 743-761.

[7] R. Benenson, M. Omran, J. Hosang, B. Schiele, “Ten years of pedestrian detection, what have we learned?”, arXiv, 2014.