Shallow-learning methods for pedestrian classification and detection in images

Master's degree in Computer Engineering for Robotics and Smart Industry
Machine Learning and Artificial Intelligence course, a.a. 2021/2022
Lorenzo Busellato[1]

[1]VR472249, lorenzo.busellato_02@studenti.univr.it

Abstract

Pedestrian detection is a key task in the computer vision field, given its obvious applications in surveillance systems, autonomous driving and robotics. This work presents a pipeline for shallow-learning based pedestrian detection. The Histogram of Oriented Gradients (HOG) method for feature extraction is described and applied to the Daimler Mono Pedestrian Classification Benchmark Data Set. Feature dimensionality reduction is performed using Principal Component Analysis (PCA). Four classification techniques, Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), Naive Bayes and a shallow neural network, are described and benchmarked with different sets of hyperparameters. The best performing method of each class is then retrained using hard-negative mining. The benchmarks are then compared against each other to determine the best performing method in terms of accuracy in classification. Finally the best performing classifier is showcased by using it in a full pipeline for pedestrian detection applied to a video sequence.

Contents

I. Introduction

THE task of pedestrian detection, i.e. the task of localizing pedestrians within an image, is one of the most tackled instances of the broader class of object detection problems. This is due to its natural applications in autonomous driving, surveillance and robotics. For instance, in the US alone about 5000 of the 35000 annual car accident fatalities involve pedestrians [1], therefore the need for automated methods for pedestrian detection is apparent.

The task of pedestrian detection presents a series of complexities regarding the high variability in scale and size of the person within the image, occlusions with the rest of the scene, susceptibility to lighting conditions and the real time detection requirement.

The main models that are used for pedestrian detection are hand-crafted models and deep learning models. Hand-crafted models are based on classification of hand-crafted features. Deep learning models on the other hand typically use convolutional neural networks for the extraction of features, therefore producing high-level semantic representations of objects. In this work the first class of models will be analyzed and implemented. Specifically, the Histogram of Oriented Gradients (HOG) method will be used to extract features from images, because it is known[2] that it is the best performing non-deep learning based feature extractor for human detection. The classification of images will then be implemented using non-deep learning methods, namely Support Vector Machines (SVMs), K-Nearest Neighbors (KNN), Naive Bayes and a shallow neural network.

The full details of the implementation can be found on the project's GitHub page[4].

The main objectives of this work are described in section II. The data set and methodology employed are described in sections III and IV. The implementation of the methods is briefly discussed in section V. The experimental evaluation is presented in section VI. The conclusions are summarized in section VII.

II. Objectives

The main objective of this work is the implementation of a pedestrian detection pipeline, which is made up of three main components: a feature extractor, a classifier and a detector.

Regarding the classifiers, the objective is to obtain a comparison between the performance of some well-known shallow learning techniques with respect to the problem of pedestrian detection, as well as the relation between hyperparameters and classifier performance.

Finally, the best performing classifier will be used to implement a detector, therefore completing the pedestrian detection pipeline. The pipeline will then be applied to a video sequence in order to showcase its performance.

III. Dataset

The data set used in this work is the Daimler Mono Pedestrian Classification Benchmark Data Set[3]. The data set consists of four separate subsets: the base set and three additional sets. The base set is divided into five subsets, three for training (named "1", "2" and "3") and two for testing (named "T1" and "T2"). All five sets contain 4800 18×36 grayscale images containing pedestrians and 5000 18×36 grayscale images not containing pedestrians. The three additional sets each contain 1200 640×280 grayscale images for the purpose of extracting additional negative samples for the hard-negative mining procedure.

Fig. 1. Example figures from the training data set. On the left a non-pedestrian sample, on the right a pedestrian sample.

In this work, data sets 1, 2 and 3 will be grouped together in a singular data set for the training of the models, as will data sets T1 and T2 for the testing.

IV. Methodology

A. Histogram of Oriented Gradients

Given an input image I, for each pixel $(x, y) \in I$ the gradients G_x and G_y are computed. This is typically done via a convolution operation:

$$G_x = \omega_x * I \qquad G_y = \omega_y * I$$

where ω_x and ω_y are the kernels associated to the convolution operation. The most common kernels used in the context of pedestrian detection are:

$$\omega_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \omega_y = \omega_x^T$$

They are the most commonly used because experimentally it is observed[2] that they produce the least amount of detected false positives per moving window (FFPW) compared to other kernels.

After the convolution operation, two gradient matrices are obtained, G_x and G_y. For each pixel $(x, y) \in I$ then the magnitude μ and angle θ are computed:

$$\mu(i, j) = \sqrt{G_x(i, j)^2 + G_y(i, j)^2}$$

$$\theta(i, j) = \left| \tan^{-1}\left(\frac{G_y(i, j)}{G_x(i, j)} \right) \right|$$

In practice the angle is computed with $atan2$, therefore it is in the range $(\pi, \pi]$.

The resulting gradient magnitude and gradient angle matrices are then divided into 8×8 patches, for each of which a 9-point histogram of oriented gradients is computed. A histogram is a series of bins labeled with angle values. The number of bins is determined by the angle step between the bins, for instance with an angle step of 20 degrees the histogram will have $180° / 20° = 9$ bins.

For each pixel in the patch, the corresponding gradient angle determines two bins on the histogram that the pixel votes for. The two bins selected are those which represent the two closest angles to the given gradient angle value. The value added to the selected bins is proportional to the distance of the gradient angle to the bin angle, the gradient magnitude and the step between the bins:

$$v_{closest} = \frac{(binAngle - \theta)\mu}{angleStep}$$

$$v_{2nd-closest} = \frac{(\theta - binAngle)\mu}{angleStep}$$

Repeating the process for each 8 × 8 patch, a 16×8 matrix of histograms is constructed. To reduce the variability in gradient intensities due to lighting conditions and foreground-background contrast, a block normalization procedure is introduced. The 16×8 histogram matrix is scanned with a 2×2 window producing 15×7 blocks, for each of which a normalized histogram vector is computed by concatenating the four histograms contained in the block and normalizing the resulting vector. Finally, the feature descriptor for the image is constructed by concatenating the normalized histogram vectors. From each 8×8 patch a 1×9 vector is collected, which is then concatenated with another three 1×9 vectors composing a 1×36 vector, which is finally concatenated with another fourteen 1×36 vectors, resulting in a 1×3780 feature vector.

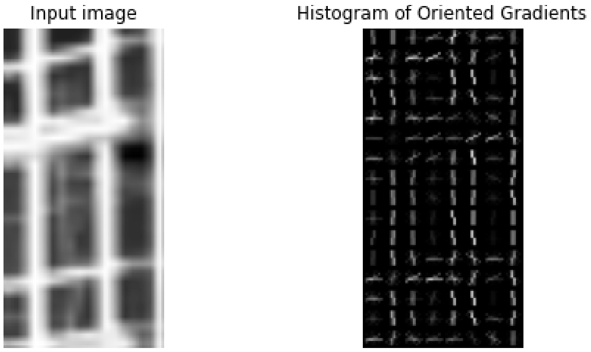

Fig. 2. Histogram of Oriented Gradients representation of a non-pedestrian sample.

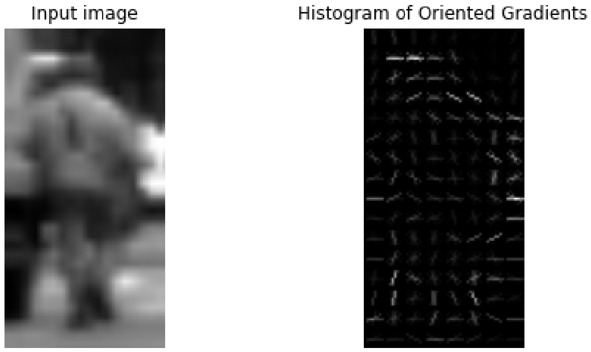

Fig. 3. Histogram of Oriented Gradients representation of a pedestrian sample.

A representation of the resulting feature vector can be seen in figures 2 and 3. The representation is obtained by plotting the 9×1 normalized histograms in each 8×8 cell. The dominant direction of the histogram is able to capture contours, especially straight lines such as the frame of the window in figure 2 or the legs of the pedestrian in figure 3. The gradient magnitudes also carry information on the lighting of the scene, with greater magnitudes corresponding to lighter areas.

B. Principal Component Analysis

Principal Component Analysis (PCA) is a technique used in order to reduce the dimensionality of the feature vectors. It is based on the idea of projecting data from an highly dimensional space to a space with lower dimension, while retaining as much information as possible. Information is encoded by the underlying variance of the data. In other words the data is projected into a space such that the first direction is the direction of maximum variance, the second direction is the direction of second most maximum variance, orthogonal to the first, and so on.

Such directions, which are called principal directions, are encoded by the eigenvectors of the covariance matrix of the data. The eigenvectors corresponding to the directions of maximum variance correspond to the maximum eigenvalues.

The algorithm is as follows. Given a set of feature vectors x^k, with $k = 1, \ldots, M$:

- Center the data around the mean:

$$x_c^k = x^k - \frac{1}{M}\sum_{k=1}^{M} x^k$$

- Compute the covariance matrix C of the centered data.
- Calculate the eigenvectors and eigenvalues of C.
- Sort the eigenvalues from largest to smallest.
- The transformation matrix T, used to reduce the data into N dimensions, is made up of the first N eigenvectors corresponding to the largest N eigenvalues.
- The transformed data is therefore:

$$\omega^k = x_c^k \cdot T$$

The choice of N is related to the amount of variance of the original data is to be kept. Typically N is chosen such that the variance kept is at least 80% of the original variance. In this work $N = 766$ was chosen in order to keep 90% of the variance of the original data.

C. Naive Bayes

Naive Bayes is a class of supervised learning methods based on the "naive" assumption that each pair of features is conditionally independent given the value of the class variable.

Given the class variable y and the dependent feature vector $x = (x_1, \ldots, x_n)$, Bayes' theorem states that:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{1}$$

Under the conditional independence assumption that for all i:

$$P(x_i \mid y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i \mid y)$$

the theorem reduces to:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, and acts only as a scaling factor:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Therefore the following classification rule can be derived:

$$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

The different naive Bayes classifier differ only with respect to the chosen shape of the distribution of $P(x_i \mid y)$. In this work only Gaussian Naive Bayes will be considered. The model assumes that the likelihood distribution of the features is Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

D. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a non-parametric supervised learning method used for classification. It is based on the assumption that feature vectors belonging to the same class are close together in the feature space. Given an unlabeled sample, the distance from it to every sample in the training set is computed using some metric, typically euclidean. The label of the sample is then given by the mode of the K samples with the smallest distance from the unlabeled sample. The algorithm is as follows:

- Initialize K as the desired number of neighbors.
- For each sample in the training data set, compute the distance between the unknown sample and the current sample and add it to a list.
- Sort the list of distances and pick the first K entries.
- The label for the unknown sample is the mode of the labels of the K entries.

Regarding the choice of K, a common heuristic is to pick $K = \sqrt{N}$, where N is the number of training samples. A more robust approach is to repeat the training of the model with varying K values, among which a "best" one is to be selected with respect to the minimization of some error metric, typically accuracy. In this work the second approach will be employed, showing that choosing the square root of the number of samples is not always the best option in terms of accuracy.

Another choice to be made is the adopted distance metric. In this work the Minkowski metric will be used. The Minkowski metric of order p, where p is an integer, between two points $X = (x_1, \dots, x_n) \in \mathbb{R}^n$, $Y = (y_1, \dots, y_n) \in \mathbb{R}^n$ is defined as:

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

For $p = 1$ the metric reduces to the so-called Cityblock metric, which is the sum of the absolute differences of the cartesian coordinates of the points. For $p = 2$ the metric reduces to the Euclidean distance. In this work the model will be employed with $p = 1, 2$, showing the differences in classifier performance in the two cases.

E. Support Vector Machines

Support Vector Machines (SVMs) are a class of supervised learning algorithms whose main idea is to find an optimal hyperplane in an n-dimensional space, with n number of features, that maximizes the distance, called margin, between the classes of data points. Such an hyperplane constitutes the decision boundary used by a classifier to classify data points, on the basis of which "side" of the hyperplane they fall in. The position and orientation of the hyperplane are influenced by the data points closer to it, which are called support vectors.

If data is not linearly separable in its original space, which is often the case,a parameter C and slcak variables ε_i are introduced. The slack variables allow for some support vectors to exceed the margin, and the parameter C sets the cost for a wrong classification. Furthermore, a transformation to an higher dimensional space is introduced in order to achieve linear separability. This transformation is called kernel. The main kinds of kernels are:

- Linear:
$$K(x, z) = x \cdot z$$
- Polynomial
$$K(x, z) = (x \cdot z + 1)^p$$
- Sigmoid:
$$K(x, z) = \tanh(ax \cdot z + b)$$
- Radial Basis Function (RBF):
$$K(x, z) = exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

F. Neural networks

G. Hard-negative mining

Hard-negative mining is a technique used to increase the accuracy of object classifiers, by increasing the number of true negatives and decreasing the number of false positives classified by an already trained classifier.

Given an image known not to contain any pedestrians, a Gaussian pyramid is first extracted. That is to say that the image is progressively downsampled, obtaining a set of images at various resolutions. Then a window is slid over each image in the pyramid, and the HOG feature vector is computed. The feature vector is classified by the classifier and, in case the classifier misclassifies the feature vector, the latter is added to the original training set.

Repeating the process on many additional images will yield a large amount of misclassified feature vectors added to the training set. By retraining the classifier on the expanded training set then its performance should increase.

H. Pedestrian detection

To detect pedestrians in a given input image, a sliding window approach is used.

Firstly, a Gaussian Pyramid is extracted from the image. That is to say the image is progressively downsampled, obtaining a set of lower resolution versions of the input image at every scale. Then, a sliding window is passed over each image in the pyramid, and a HOG feature vector is computed for each step of the sliding. Finally, the feature vector is classified by one of the previously described classifiers. If the classifier identifies the window as a pedestrian, the coordinates of its top left corner as well as the confidence score and its properly scaled width and height, define the bounding box around the pedestrian.

This approach, while effective in locating pedestrians within an image, produces multiple proposals for the bounding box. This is because windows that contain only a portion of a pedestrian still can produce feature vectors similar to those produced by windows that fully capture a pedestrian.

I. Non-maximum suppression

Non-maximum suppression (NMS) is a technique for the filtering of predictions made by an object detector. An object detector typically produces a list B of proposed bounding boxes along with the confidence score S assigned to each of them. The algorithm is as follows:

- Select from B the proposal with the highest confidence score, remove it from B, and add it to an initially empty list D.
- Compare the proposal with all other proposals in D by computing the Intersection Over Union (IOU):
$$IOU = \frac{\text{Intersection Area}}{\text{Total Area}}$$
- Remove from B all the proposals with an IOU greater than some threshold.
- Repeat steps 1-3 until B is empty.

The list D then contains only the bounding boxes that have the highest confidence score and that intersect the least with other bounding boxes already in D.

V. Implementation

A. Feature extraction

B. Classification

C. Detection

VI. Experiments and results

A. Adopted metrics

To evaluate the performance of the various classifiers, the following metrics were used. Given TP the number of true positives, TN the number of true negatives, FP the number of false positives and FN the number of false negatives detected by the classifier:

- Accuracy:
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$
Accuracy is a raw evaluation of the number of correct guesses with respect to the total number of samples. It is well suited for binary classification problems such as pedestrian classification, but it can be misleading for the detection phase since in an image the pedestrian labels are typically sparse.
- Precision:
$$Precision_P = \frac{TP}{TP + FP}$$
$$Precision_N = \frac{TN}{TN + FN}$$
Precision is the proportion of predicted positives (or negatives) that actually are positive (or negative) samples. It is an important metric for pedestrian detection because the pipeline needs to limit as much as possible wrong detections, which could be problematic for applications like autonomous driving.
- Recall:
$$Recall_P = \frac{TP}{TP + FN}$$
$$Recall_N = \frac{TN}{TN + FP}$$
Recall is the proportion of actual positives (or negatives) correctly classified as positive (or negative). It is an important metric for pedestrian detection because the pipeline needs to correctly classify as many pedestrians as possible in a given image (hopefully all of them).
- F_1 score:
$$F_1 = 2\frac{Precision_P \times Recall_P}{Precision_P + Recall_P}$$
F_1 offers a trade-off between precision and recall. It is a perfectly suited metric for pedestrian detection because the pipeline needs to both maximize the detection rate, but also be sure that the detections actually are pedestrians, or in other words the higher the F_1 score, the better. For these reasons only the F_1 score related to the pedestrian label is calculated.

Since its purpose is mostly that of showcasing the best performing classifier, the detector will not be subject to any metric evaluation.

B. Performed experiments

Regarding Naive Bayes, 50 models were trained and tested with 50 different values for the smoothing parameter. This was done in order to determine what value of the smoothing parameter produces the best performing model.

Regarding KNN, 20 models were trained and tested, with 10 models being trained with Cityblock distance metric and 10 models being trained with Euclidean distance metric. In both cases the 10 models correspond to 10 different K values in the range 1-101, with N number of samples. This was done in order to plot the accuracy-K curve and thus determine what K value results in the best performing model.

Regarding SVM, 28 models were trained and tested, 7 for each kernel type (linear, poly, sigmoid and RBF). The 7 models for each kernel type correspond to 7 different values for the regularization parameter C, in the range 0.001-1000. Once again this was done in order to establish what value for C results in the best performing model.

Regarding the neural network, 24 models were trained and tested. 3 models were trained with a single layer, of sizes $(2500,)$, $(2000,)$ and $(1250,)$ respectively, while 3 models were trained with two layers, of sizes $(2500, 1250)$, $(2000, 1000)$ and $(1250, 625)$ respectively. In both cases all 3 models were trained with 4 different values in the range 0.0001-0.1 for the regularization parameter α. Since shallow neural networks are not typically used to classify HOG features, this was done to obtain an indication of the number of hidden layers, the number of neurons in each layer and the value for α that produce the best model.

To further improve performance, the best classifier of each class was then retrained by applying hard-negative mining.

C. Results

VII. Conclusions

References

[1] Umesh Shankar. *Pedestrian roadway fatalities*. Tech. rep. 2003.

[2] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[3] Stefan Munder and Dariu M Gavrila. “An experimental study on pedestrian classification”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.11 (2006), pp. 1863–1868.

[4] *mlai_project*. https://github.com/lbusellato/mlai_project.