Shallow learning methods for pedestrian classification and detection

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

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 Pedestrian Detection Benchmark Dataset. Three classification techniques, Support Vector Machines (SVMs), K-Nearest Neighbors (KNN) and Parzen Windows, are described and benchmarked with different sets of hyperparameters. The benchmarks are then compared against each other to determine the best performing method in terms of accuracy in detection. Finally the best performing SVM is showcased by applying it to a video sequence.

CONTENTS

I Introduction 1

II Objectives 1

III Dataset 1

IV Methodology 1
- IV-A Feature extraction 1
- IV-B K-Nearest Neighbors 2
- IV-C Parzen Windows 2
- IV-D Support Vector Machine 2
- IV-E Pedestrian detection 2
- IV-F Non-maximum suppression 2

V Results 2
- V-A Adopted metrics 2
- V-B Results 2

VI Conclusions 2

References 2

LIST OF FIGURES

LIST OF TABLES

I Measured metrics for the tested KNNs 3
II Measured metrics for the tested SVMs 3

Shallow learning methods for pedestrian classification and detection

I. Introduction

Pedestrian detection, i.e. the task of detecting and 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 [2] [3] [4]. 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 [5] [6] [7].

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. The classification of images will then be implemented using non-deep learning methods, namely Support Vector Machines (SVMs), K-Nearest Neighbors (KNN) and Parzen Windows. Finally the best performing classifier will be used to implement a pedestrian detector.

The main objectives of this work are described in section II. The dataset and methodology employed are described in sections III and IV. The experimental evaluation is presented in section V. The conclusions are summarized in section VI.

II. Objectives

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

The feature extractor chosen for this task is the HOG, given its improved performance with respect to other common feature extractors employed in image processing [8].

Regarding the classifier, three models will be implemented: SVMs, KNN and Parzen Windows. All models will be trained with different combinations of hyperparameters in order to obtain a benchmark that will allow for the comparison between different instances of the same model as well as the comparison with the other models.

The best performing classifier, taking into account execution times, will then be applied to a video sequence in order to realize a pedestrian detector.

III. Dataset

The dataset used in this work is the Daimler Pedestrian Detection Benchmark Dataset [9]. The set contains 15560 48x96 grayscale image samples of pedestrians, to be used as positive samples, and 6744 640x480 grayscale images not containing pedestrians, to be used for the extraction of negative samples. From each image in the non-pedestrian dataset, 6 randomly picked 64x128 image patches are extracted, for a total of 40464 negative samples. Of both subsets, 80% of samples will be kept as training data, while the rest will be used as testing data to validate the classifiers.

IV. Methodology

A. Feature extraction

For a given input image or image patch I, the first step is the conversion to grayscale and the resizing to a width of 64 pixels and an height of 128 pixels.

Then, for each pixel $(i,j) \in I$ the gradients G_x and G_y in the x and y directions are computed. This is done with a convolution operation:

$$G_x = \omega_x * I \quad G_y = \omega_y * I \tag{1}$$

where ω_x and ω_y are the kernels associated to the convolution operation. [8] has shown that the best performing kernel in terms of detected false positives per moving window (FFPW) is the monodimensional:

$$\omega_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \omega_y = \omega_x^T \tag{2}$$

After the convolution operation, two 128×64 gradient matrices G_x and G_y are obtained. For each pixel (i,j) in the image, the magnitude μ and angle θ are then computed:

$$\mu(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{3}$$

$$\theta(i,j) = \left| \tan^{-1}\left(\frac{G_y(i,j)}{G_x(i,j)}\right) \right| \tag{4}$$

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:

$$v_{closest} = \frac{(binAngle - \theta)\mu}{angleStep} \tag{5}$$

$$v_{\text{2nd-closest}} = \frac{(\theta - binAngle)\mu}{angleStep} \tag{6}$$

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 $15 \cdot 7$ 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.

B. K-Nearest Neighbors

K-Nearest Neighbors 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:

1) Initialize K as the desired number of neighbors.
2) For each sample in the training dataset, compute the distance between the unknown sample and the current sample and add it to a list.
3) Sort the list of distances and pick the first K entries.
4) 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 (e.g. 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 $X = (x_1, \ldots, x_n) \in \mathbb{R}^n, Y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ 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 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 Minkowski metric with $p = 1, 2, 3$ will be used.

C. Parzen Windows

D. Support Vector Machine

E. Pedestrian detection

To detect pedestrian in a given input image, a sliding window approach is used. First a Gaussian Pyramid is extracted from the image. That is to say the image is progressively downsampled, obtaining a set of representations 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 windows that capture a pedestrian fully.

F. 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:

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

V. Results

A. Adopted metrics

B. Results

VI. Conclusions

References

[1] U. Shankar, "Pedestrian roadway fatalities," Tech. Rep., 2003.

TABLE I
MEASURED METRICS FOR THE TESTED KNNS

metric	K	A	P_{NP}	P_P	R_{NP}	R_P
cityblock	61	96.695	0.957	0.997	0.999	0.884
cityblock	111	95.768	0.946	0.995	0.998	0.853
cityblock	161	95.261	0.940	0.994	0.998	0.836
cityblock	211	94.913	0.936	0.994	0.998	0.823
cityblock	261	94.584	0.932	0.994	0.998	0.811
cityblock	311	94.334	0.929	0.993	0.998	0.803
cityblock	361	94.174	0.927	0.991	0.997	0.798
euclidean	61	83.341	0.812	0.999	0.999	0.403
euclidean	111	80.205	0.785	0.998	0.999	0.291
euclidean	161	78.432	0.770	0.997	0.999	0.228
euclidean	211	77.461	0.762	0.997	0.999	0.193
euclidean	261	76.873	0.757	0.996	0.999	0.172
euclidean	311	76.392	0.753	0.996	0.999	0.155
euclidean	361	76.151	0.752	0.996	0.999	0.146
minkowski	61	74.601	0.739	0.999	0.999	0.090
minkowski	111	73.194	0.729	0.999	0.999	0.039
minkowski	161	72.766	0.726	0.999	0.999	0.024
minkowski	211	72.490	0.724	0.999	0.999	0.014
minkowski	261	72.365	0.723	0.999	0.999	0.010
minkowski	311	72.285	0.722	0.999	0.999	0.007
minkowski	361	72.249	0.722	0.999	0.999	0.005

TABLE II
MEASURED METRICS FOR THE TESTED SVMS

kernel	C	A	P_{NP}	P_P	R_{NP}	R_P
linear	0.001	99.46	0.998	0.986	0.994	0.995
linear	0.01	99.64	0.997	0.995	0.998	0.992
linear	0.1	99.67	0.997	0.995	0.998	0.993
linear	1	99.64	0.997	0.994	0.998	0.993
linear	10	99.64	0.997	0.994	0.998	0.993
poly	0.001	98.74	0.999	0.958	0.983	0.998
poly	0.01	99.72	0.998	0.996	0.998	0.995
poly	0.1	99.87	0.999	0.998	0.999	0.997
poly	1	99.88	0.999	0.998	0.999	0.997
poly	10	99.88	0.999	0.998	0.999	0.997
sigmoid	0.001	83.32	0.873	0.718	0.899	0.663
sigmoid	0.01	79.84	0.828	0.685	0.909	0.513
sigmoid	0.1	78.00	0.806	0.664	0.916	0.429
sigmoid	1	77.76	0.803	0.660	0.917	0.417
sigmoid	10	84.34	0.907	0.699	0.872	0.769
rbf	0.001	88.51	0.999	0.709	0.841	0.999
rbf	0.01	98.47	0.999	0.949	0.979	0.999
rbf	0.1	99.79	0.999	0.995	0.998	0.997
rbf	1	99.89	0.999	0.999	0.999	0.997
rbf	10	99.91	0.999	0.999	0.999	0.998

[2] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[3] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," 2009.

[4] P. Dollár, 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, pp. 743–761, 2011.

[5] Y. Pang, J. Cao, and X. Li, "Learning sampling distributions for efficient object detection," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 117–129, 2016.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[7] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[8] 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)*, vol. 1, 2005, pp. 886–893 vol. 1.

[9] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.